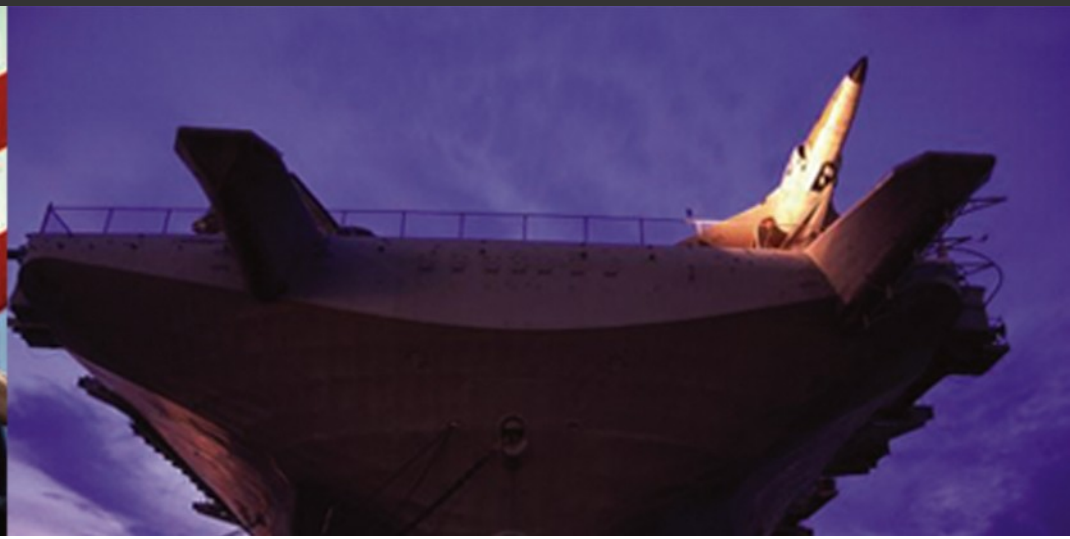
SERVICE-ORIENTED ARCHITECTURES
An Overview

Victor Harrison, CSC



CSC
EXPERIENCE. RESULTS.

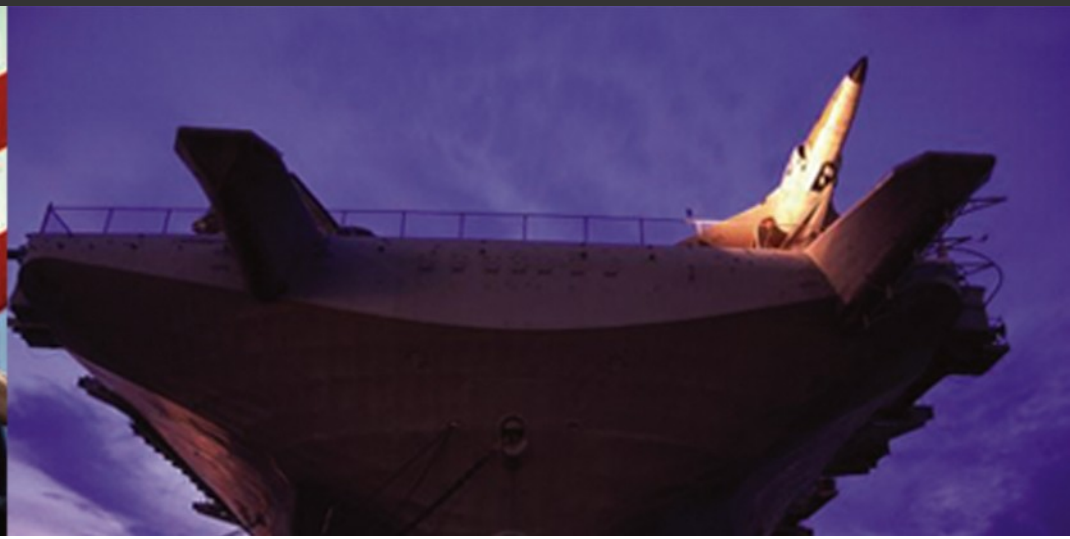
CSC
EXPERIENCE. RESULTS.

Today's Agenda

- What, exactly, is a Service Oriented Architecture
- SOA Value Proposition
- How do SOA's Work
  - The Basic Model
  - SOA and Security
- How SOA relates to DoD Architectures and Initiatives
- Lessons Learned
- Skills and Roles
- The Future

What, exactly, is a Service Oriented Architecture

And why is it so confusing



EXPERIENCE. RESULTS.

EXPERIENCE. RESULTS.

No wonder SOA is so confusing

- There are Hundreds of SOA Definitions. Some emphasize –
  - Behaviors (*A SOA delivers late binding and loosely coupled …*)
  - Componentry (*A SOA utilizes service registries, Interfaces, …)*
  - Standards (*a SOA is a set of web services that* …)
  - So on and so on and so on…
- Depending upon Point-of-View, Services of a SOA can be—
  - Pure business capabilities (Removing snow or ingesting a signal)
  - Software things (the Registry Service , logon Service, etc.)
  - Infrastructure things (amount of disk, CPUs, or networks)

CSC
EXPERIENCE. RESULTS.

Beware the SOA Myths

- SOA depends upon specific architecture products or methodologies
  - Adoption of Web Services is not a synonym for successful SOA enablement
  - Seeking to implement an “Enterprise Service Bus” is not a synonym for successful SOA enablement.
- SOA implementation depends upon specific enabling products
  - *The objective is not to make soap suds, its to wash clothes*.
  - Enabling environments and tools are chosen based upon SOA specifications, not the other way around
  - Do not be fooled into thinking that usage of a specific toolset means success.
- Delivery of a Homogeneous set of Foundational Services is Possible

The Reality: SOA is a specification for desirable behaviors such a interoperability, integration, successful execution in a heterogeneous environment, and pure marketplace driven reuse

EXPERIENCE. RESULTS.

Adopting the OMG's Harmonized Definition

- a SOA is *an **architectural style** for a community of providers and consumers of services to achieve mutual value, that:*
  - *Allows participants in the communities to work together with **minimal co-dependence** or technology dependence*
  - *Specifies the **contracts** to which organizations, people and technologies must adhere in order to participate in specific communities*
  - *Provides for business value and **business processes** to be realized by the community*
  - *Allows for a **variety of technologies** to be used to facilitate interactions within the community"*

But does this answer the question:
How do you know what a good SOA is?

CSC
EXPERIENCE. RESULTS.

SOAs are the result of dynamic and not static

Capability	Description and Design Principles	How Applied
Late binding	The resolution of a service request as late as possible during execution.	Service Proxies; reflexive bindings, service registries, COIs
Loose coupling	The low degree of mutual interdependence between components.	Data adapters, SOAP messages, generalized Metadata repositories
Discrete functions	The specification of each method of a component is independent of all other methods.	SPRINT planning based upon which operations are to be delivered, when.
Service Discovery	Query, browse, offer, and select operations for specifying operations to satisfy a specific request.	Short term: Service Proxies; long term: "behavioral browsing", reflection.
Autonomous deployment	The ability to deploy components and interfaces into production with minimal constraints on how, when, or how often the deployment occurs.	Our SCRUM-based delivery process. Usage of Service Delegates, Facades, and Proxies.
Message-Based Integration and Interaction	The usage of event-generated messages from autonomous components to dynamically form application contexts; Support for mediated and non-mediated messaging from all types of connected services	SOAP based message formats for the data packets; deep crawling; COIs and application specific 'ontologies'; Support for mediated and non-mediated synch, asynch & pub-sub messaging.
implementation- neutral	Not associating a service with a specific set of programming languages or implementations	Interfaces abstraction, environment dispatching between enclaves; SAML
Coarse grained components	Design and selection of enabling components represent classes of business entities.	Product-centric nature of delivery –each IDP deliverable is a product.
Dynamic	Dynamic formation of a process or a service	Usage of Orchestration (BPEL) to enable

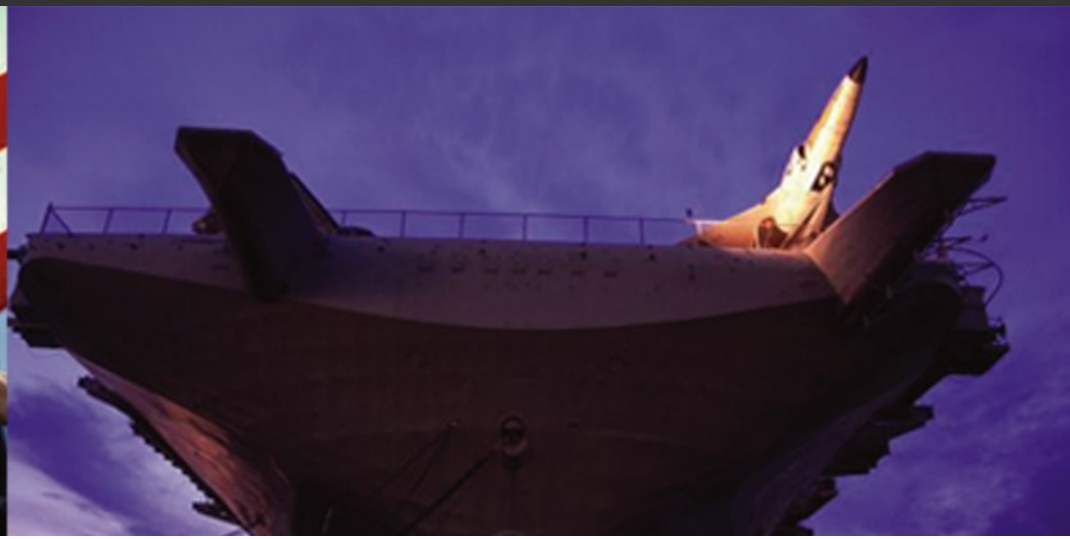
CSC
EXPERIENCE. RESULTS.

SOAs enable emergence and *conjunctive*

Capability	Description and Design Principles	How Applied
Service contracts	The basis for finding and delivering a service that 'fits' by containing a detailed specification of actions to be performed and data to be provided.	Metadata defined application contexts represented by model for information collection and COI specification.
Dynamic Process Formation	Processes are created as dynamic service collaborations (above) that result in time-dependent sequences that satisfy specific objectives.	Dynamic (temporal and event driven) as well as pre-defined process support via metadata description,
Composite services	Associations of services in a stateless and peer-to-peer collaboration to satisfy a specific capability.	Best-practice formation of services to deliver IDP-specified capabilities
Application-neutral design	Services are not designed to presume a specific application context, but have application context imported as metadata during execution.	Usage of Facades, Delegates, Factories, and Containers; Interface Facades brokers.
Consumer & Provider Metadata	Metadata driven access/usage needs (consumer metadata) matched to provider description of service (provider metadata).	Session objects of metadata. Ontologies registered in service registry, Metadata usable by all services.
Concurrently developed	Components, interfaces, and data are all capable of being developed in parallel.	Value-drive and feature based SPRINT planning with concurrent execution.
Interoperable in a heterogeneous environment	Components (1) exchange and use self-describing data (2) discover and then accept/post operational requests to components irrespective of how or where the components are enabled.	Usage of Service Proxies, Adapters, Dispatching Services between enclaves, SAML-based entitlement and attribute access; dynamic and static COIs.
Reusable	Reuse is a measure of shared elements by, minimally, reuse of a service interface's	Support for design time reuse (e.g., shared library of source and linkable objects as well

SOA Value Proposition

This is actually the hardest part of moving to SOA:
Proving that the view is worth the climb



EXPERIENCE. RESULTS.

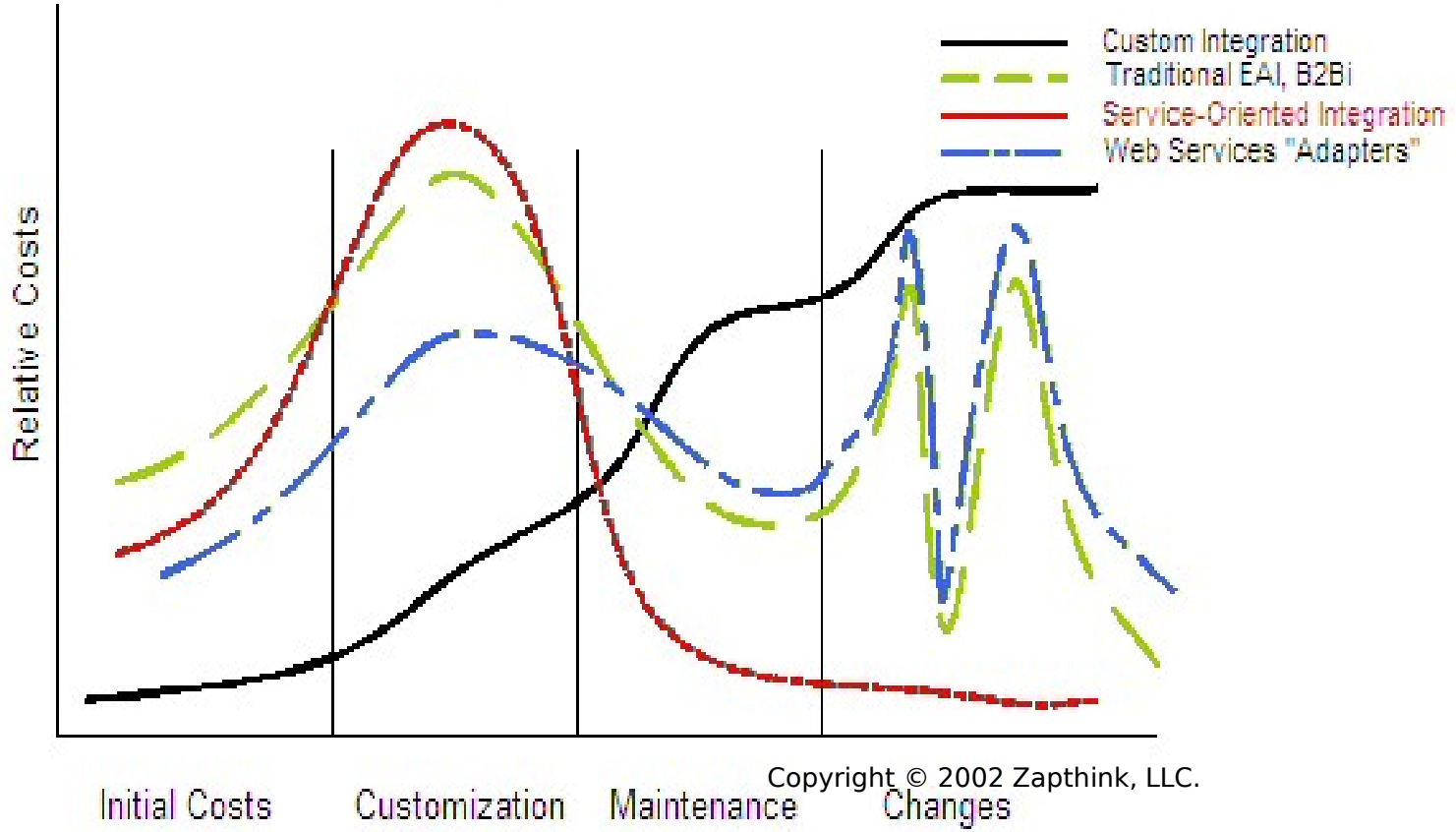
EXPERIENCE. RESULTS.

Agility

- Responsiveness
  - The ability to field features more quickly than the requirements change cycle
- Emergence
  - The ability to combine services to form new services and processes with minimal to no preconditions
    - Assemble and reassemble applications from reusable services
    - Rapid integration with mission partners

EXPERIENCE. RESULTS.

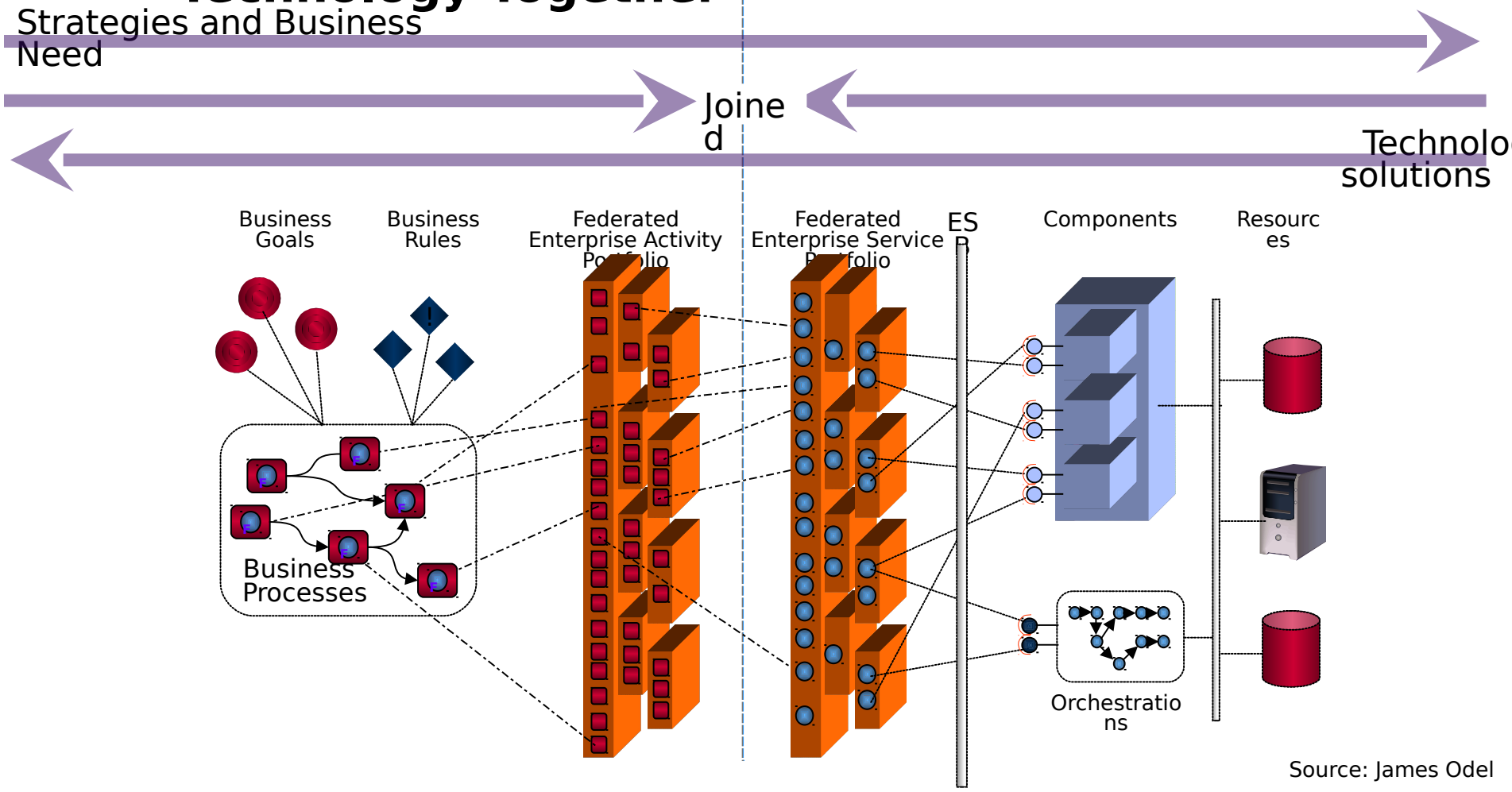
Reduced Integration Expense



Copyright © 2002 Zapthink, LLC.

CSC
EXPERIENCE. RESULTS.

Bringing the Business/Mission and the Technology Together

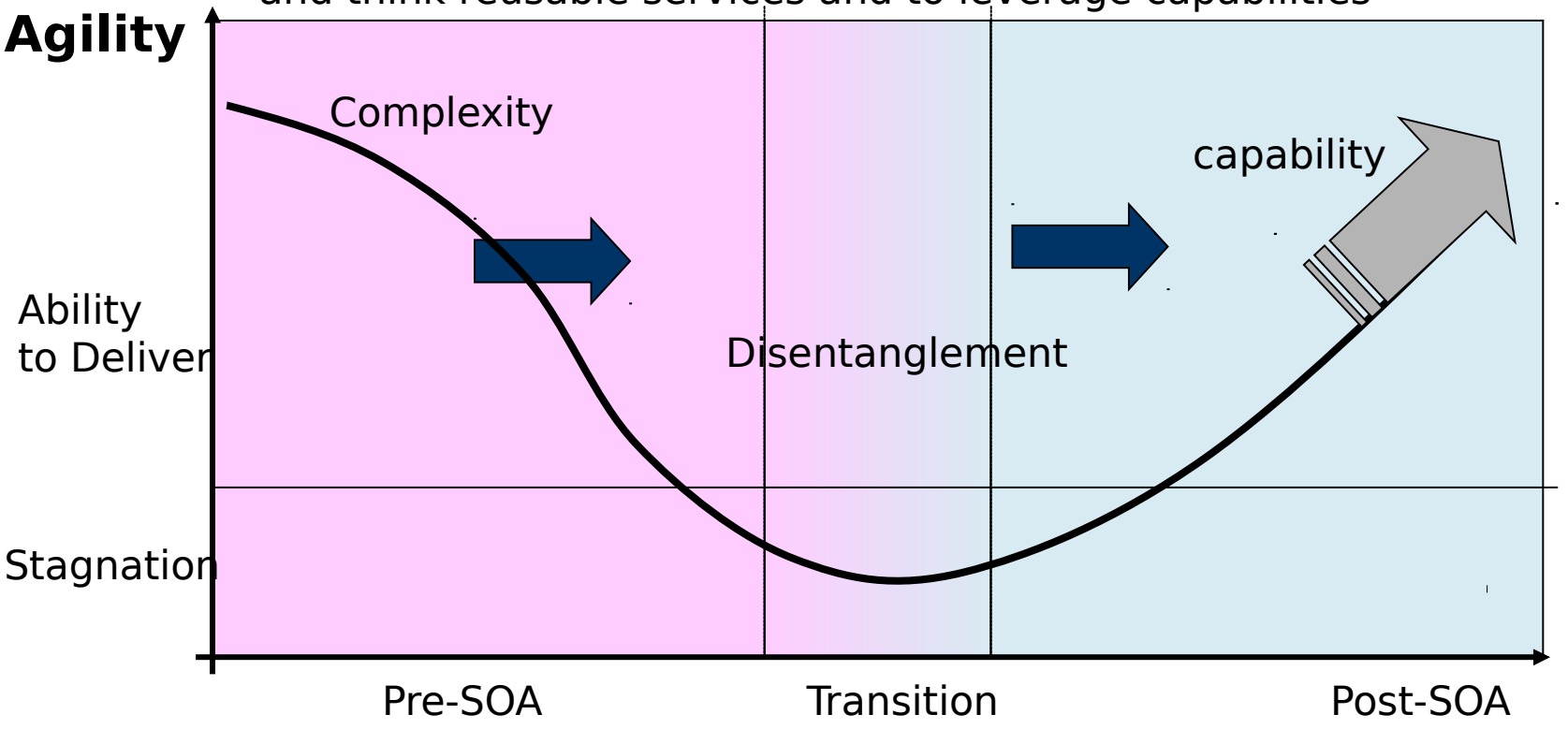


Source: James Odel

EXPERIENCE. RESULTS.

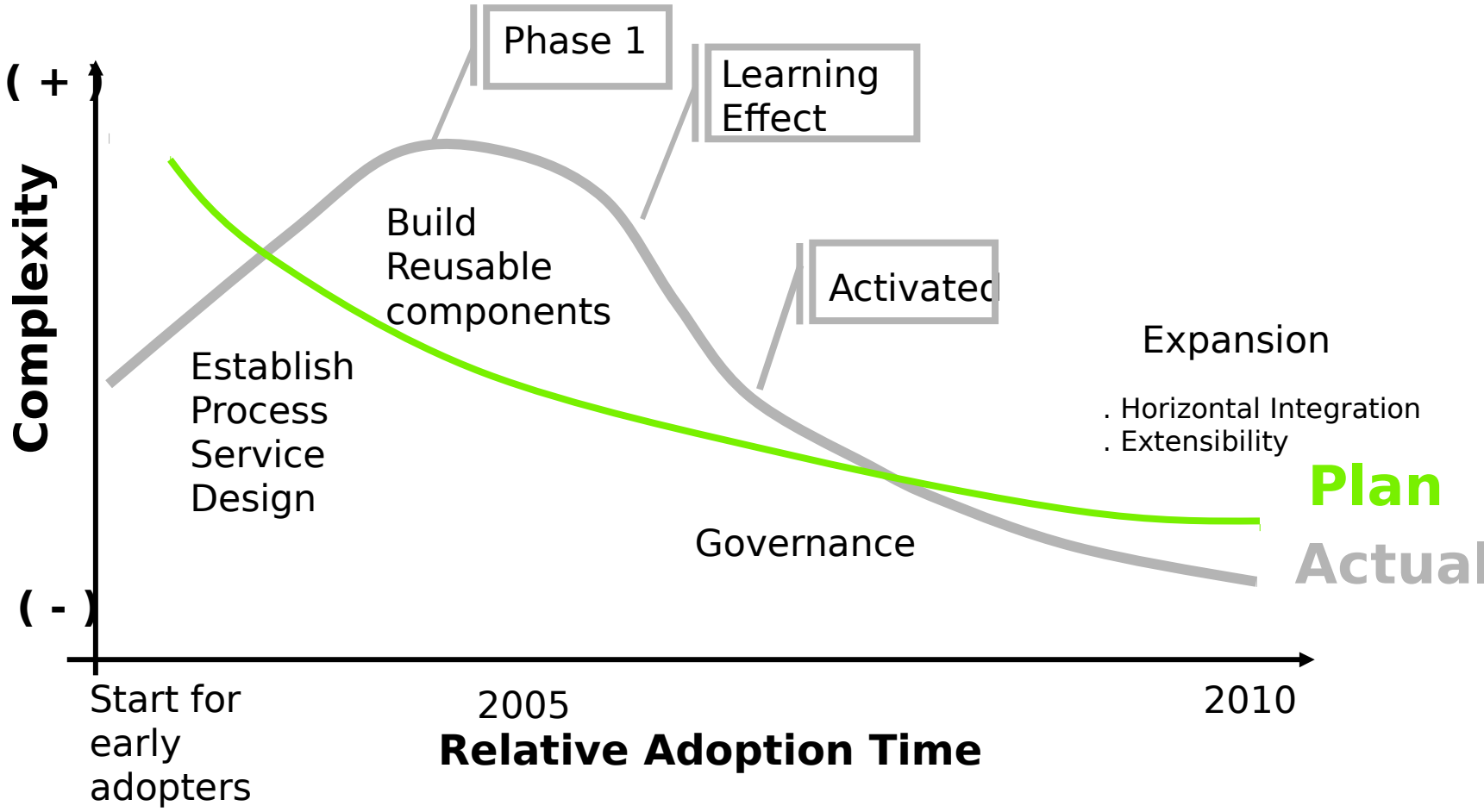
Business Transformation

SOA involves transforming the organization and IT to deliver and think reusable services and to leverage capabilities



EXPERIENCE. RESULTS.

Using SOA to Manage Complexity



EXPERIENCE. RESULTS.

Business Transformation (cont.)

- Market-Centric
  - SOA enables the Component Marketplace (Provides a mechanism for integration)
  - buy (build) Components and assemble into Systems
- Flexibility
  - Significantly expand the available pool of trading partners
- Continually Improving Execution –
  - Business/Mission Model Driven
  - Netcentricity: Shared services and behaviour
  - Use the latest services as they are created (flexible extension)

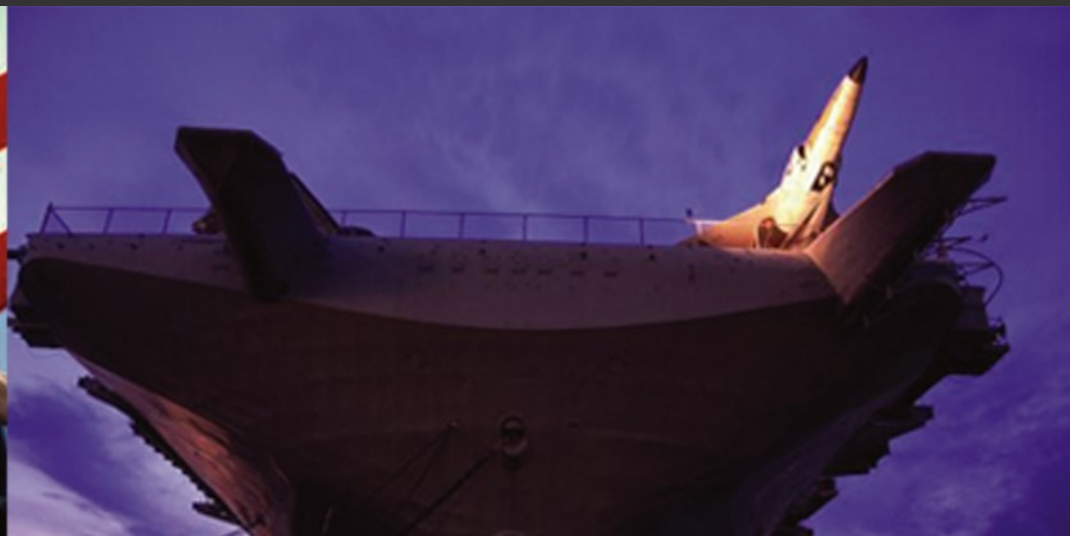
CSC
EXPERIENCE. RESULTS.

Availability, Reliability, COOP

- Reduced Business Interruptions
  - Loose coupling improves availability and recovery for distributed systems
  - Core set of thoroughly tested, highly-available services
- Improved end-user performance – especially on complex operations
  - Concurrent execution reduces elapse time
  - “Separation of Concerns” simplifies feature implementation and reduces thread execution time
  - Usually improved data access performance across a network
- Improved COOP and Fault Tolerance

How do SOA's Work: The Basic Model

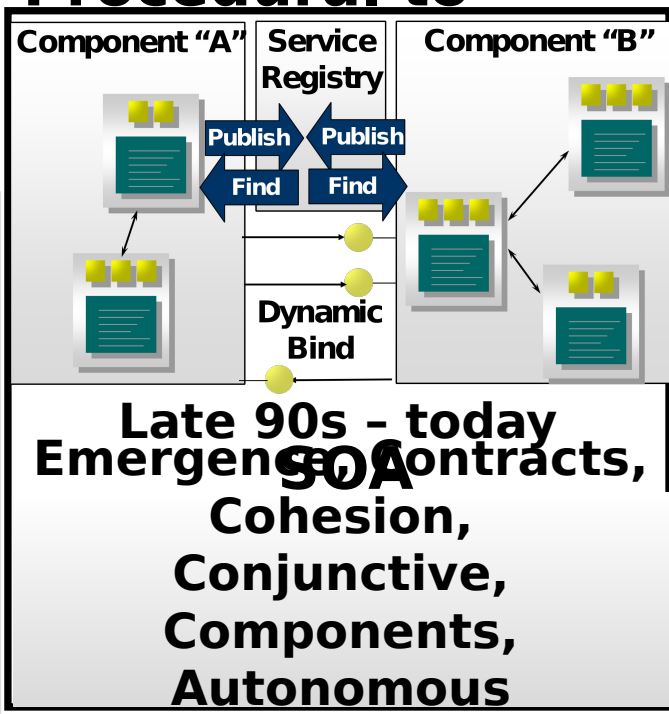
What should exist in a working SOA



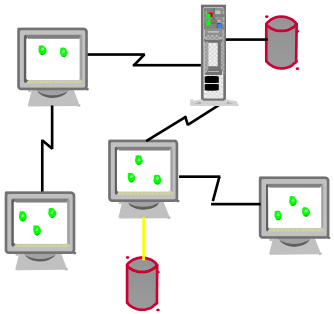
EXPERIENCE. RESULTS.

CSC
EXPERIENCE. RESULTS.

To understand how SOA works is to understand the evolution from Procedural to SOA Computing



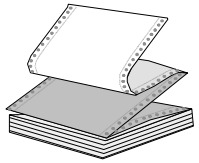
Late 90s – today
Emergence of Contracts, SOA Cohesion, Conjunctive, Components, Autonomous Applications, Distributed Applications, Objects, Interfaces



90s – today
Distributed

80s
Interactive

70s
Online



60s
Batch

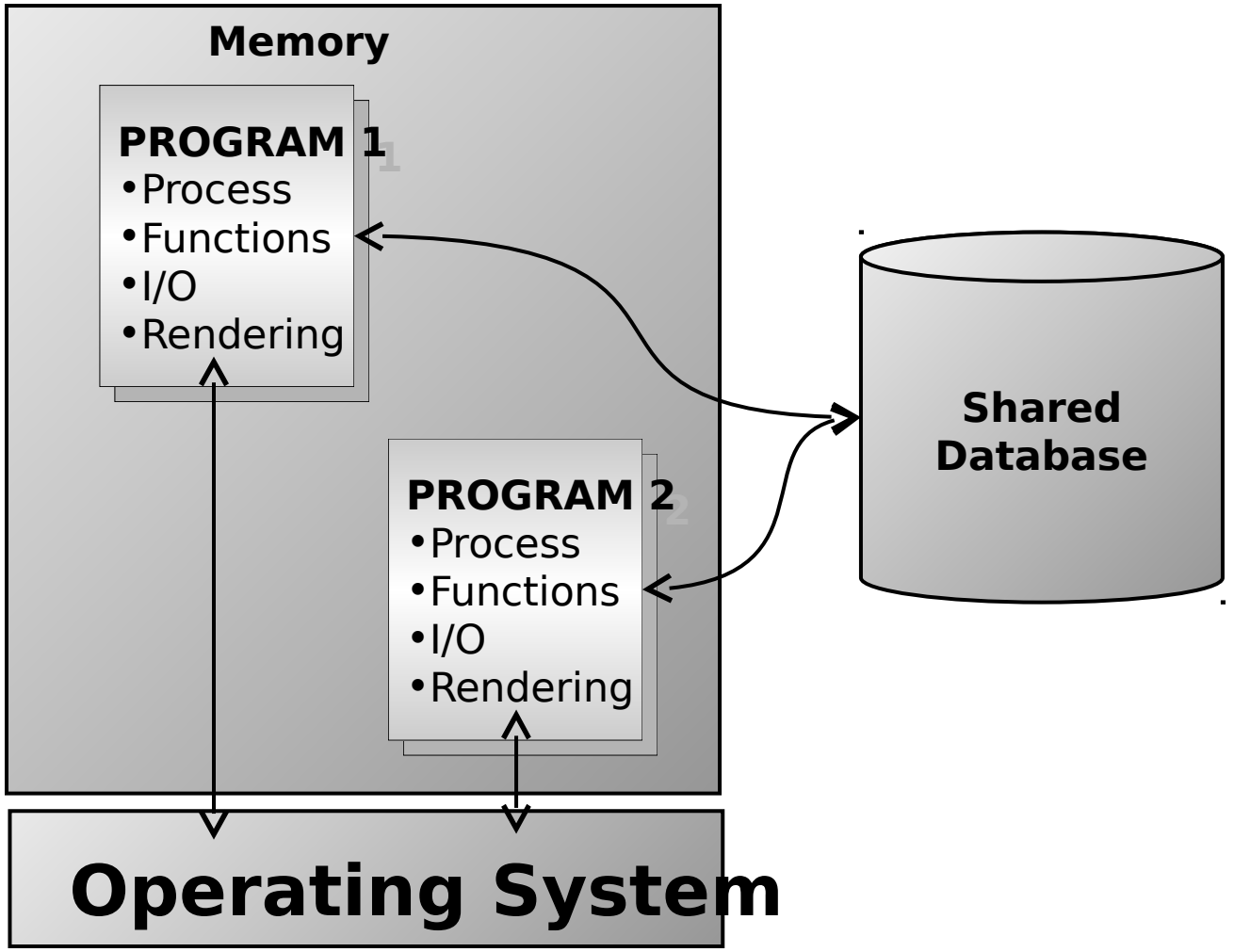
Layers/Tiers, Workflow, Session

Transactions, Unit of Work, Events

Data Separated from Program, Structure

EXPERIENCE. RESULTS.

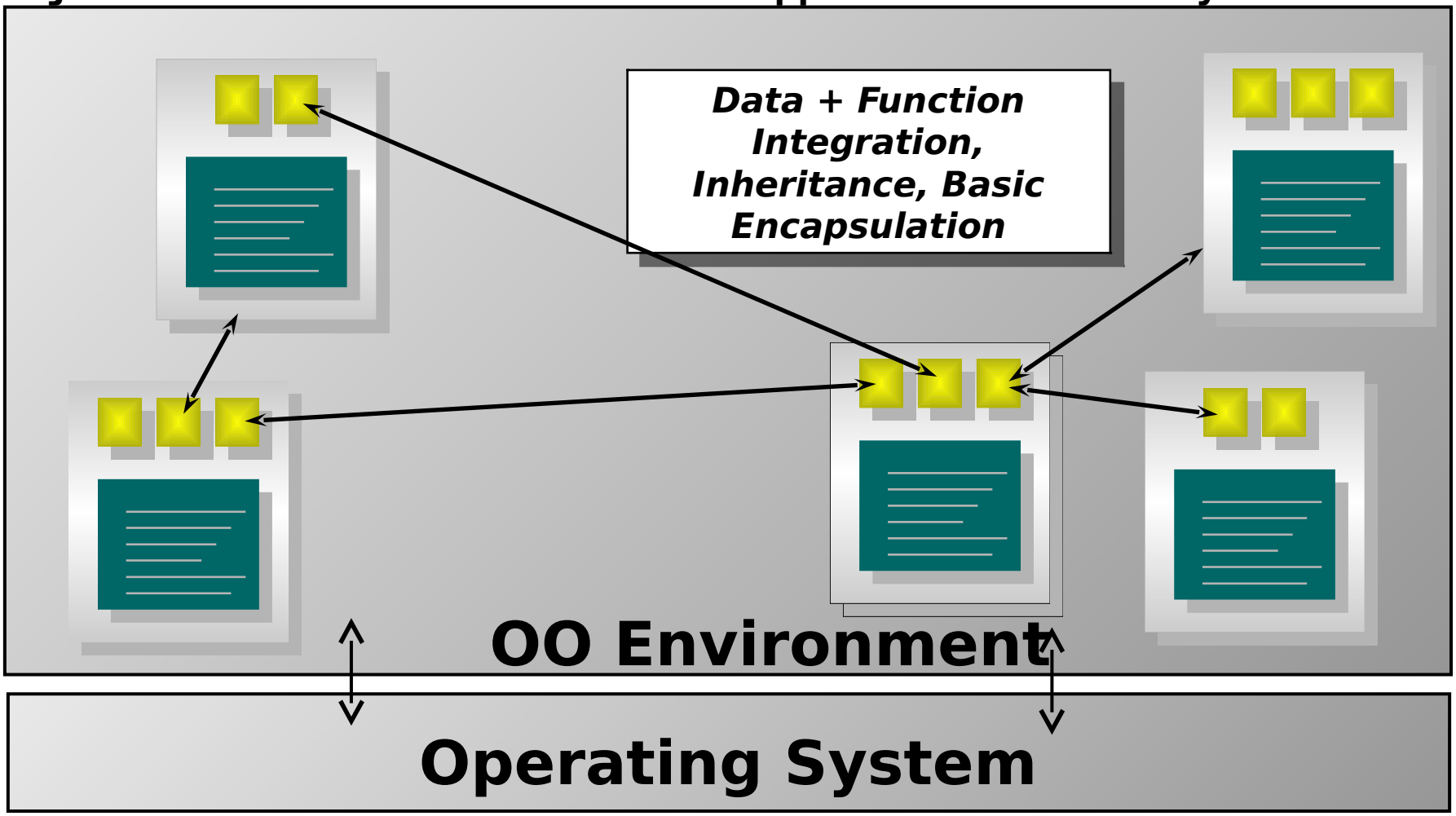
Traditional Procedural Systems dominated Batch, Online, and Interactive Systems



EXPERIENCE. RESULTS.

Evolving from Procedural to OO-based Systems

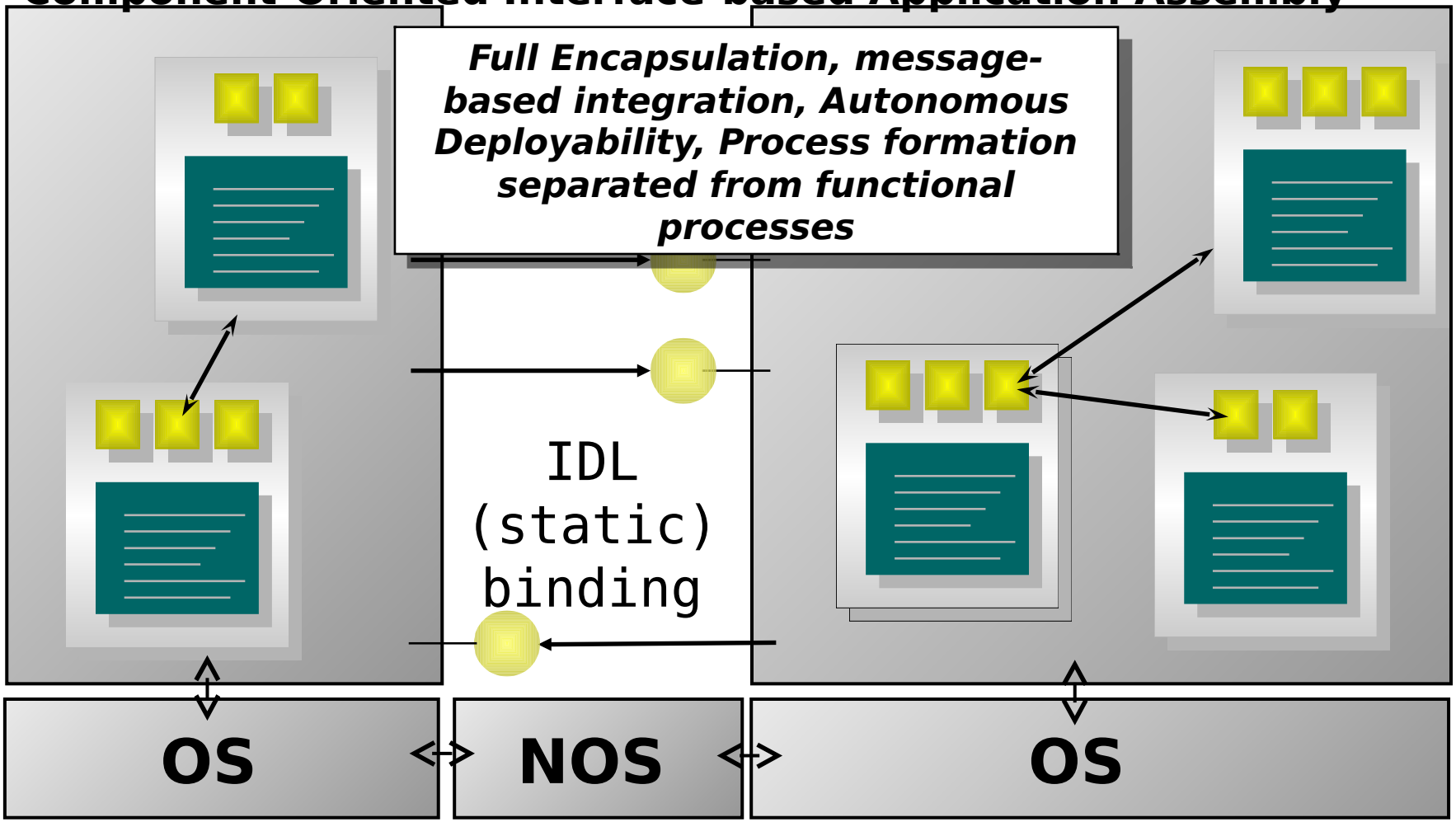
Object-Oriented Interface-based Application Assembly



CSC
EXPERIENCE. RESULTS.

Evolving from OO Systems to Interface-based Component Systems

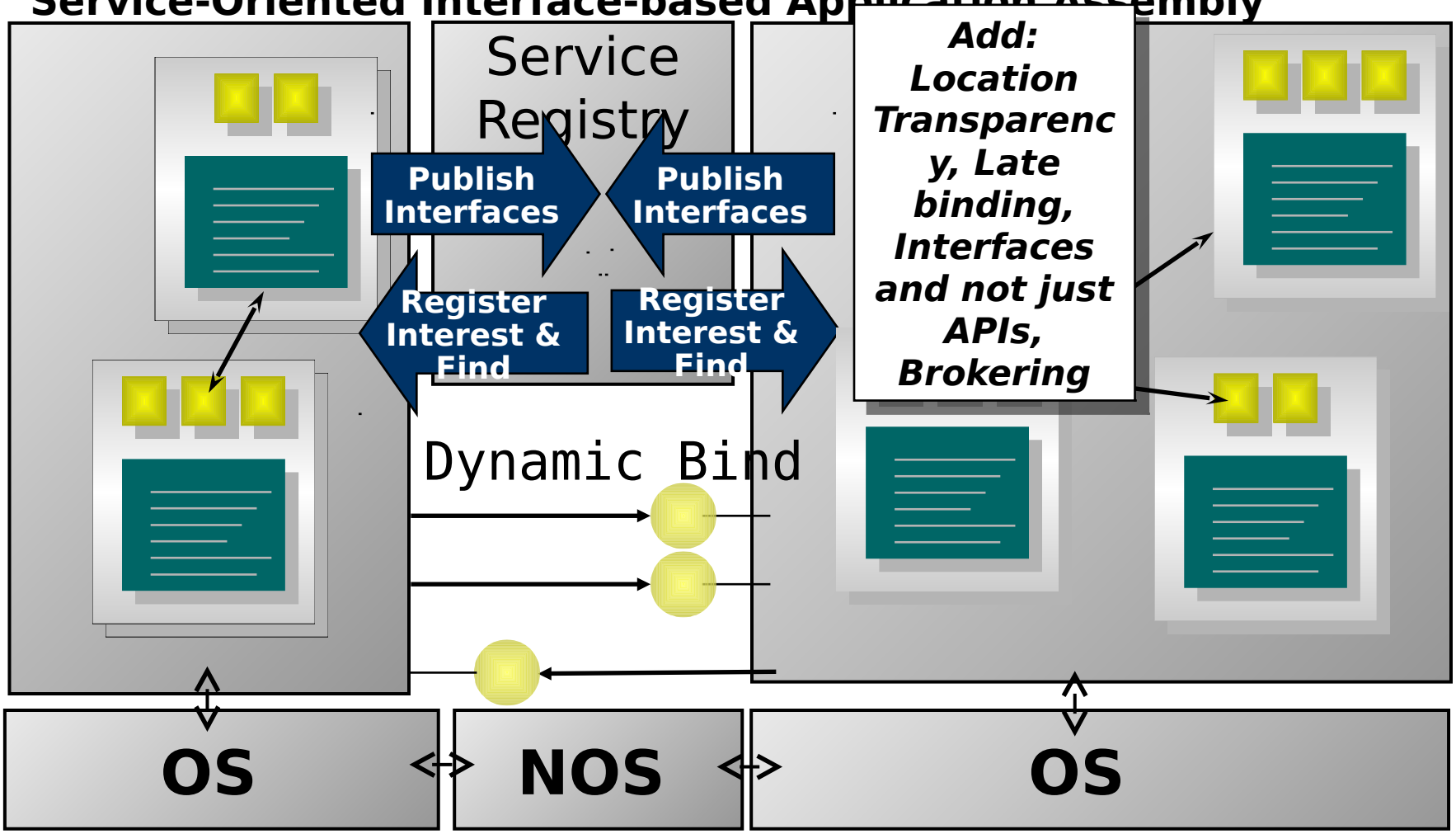
Component-Oriented Interface-based Application Assembly



EXPERIENCE. RESULTS.

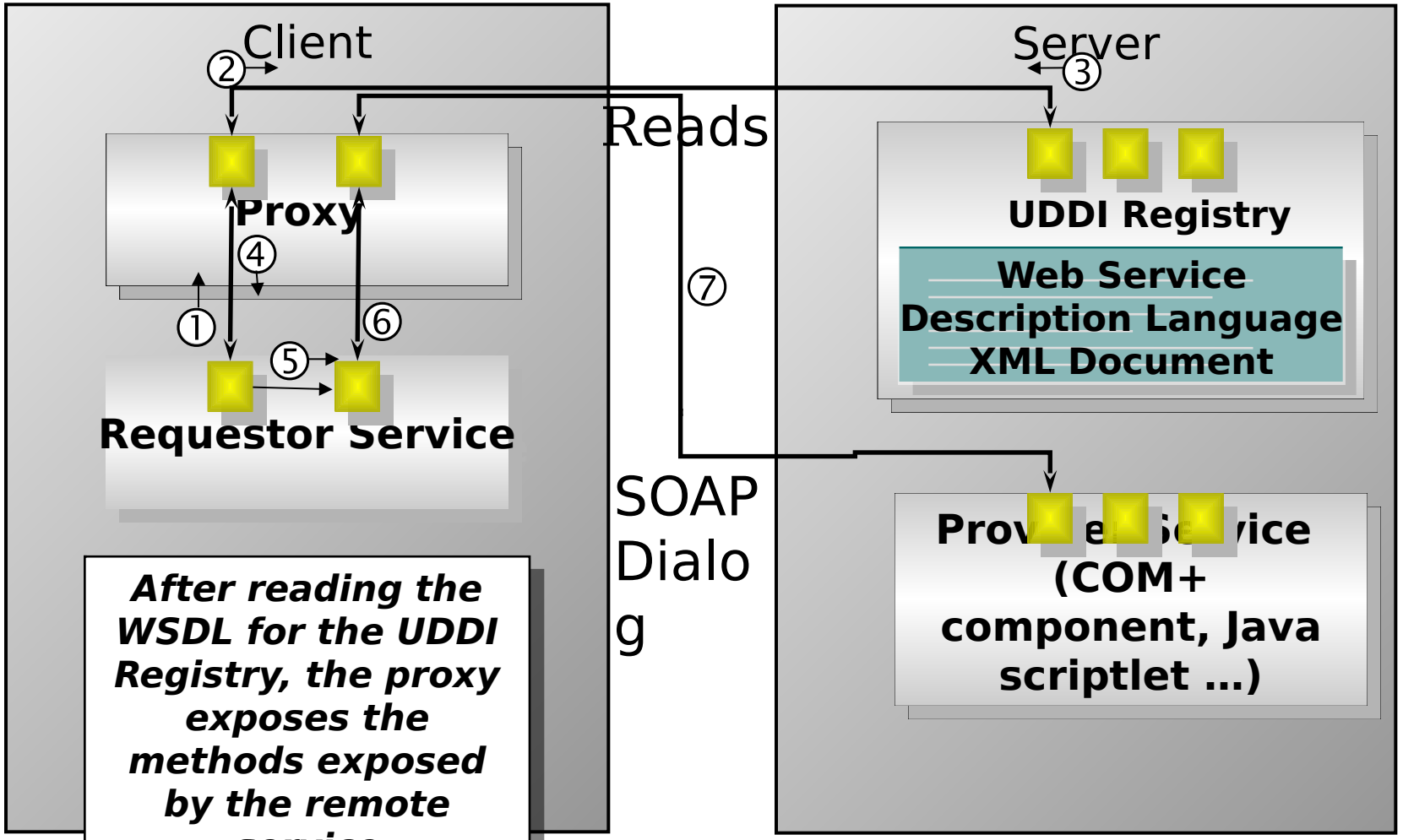
Evolving from Interface-based Component Systems to Service Oriented Systems

Service-Oriented Interface-based Application Assembly



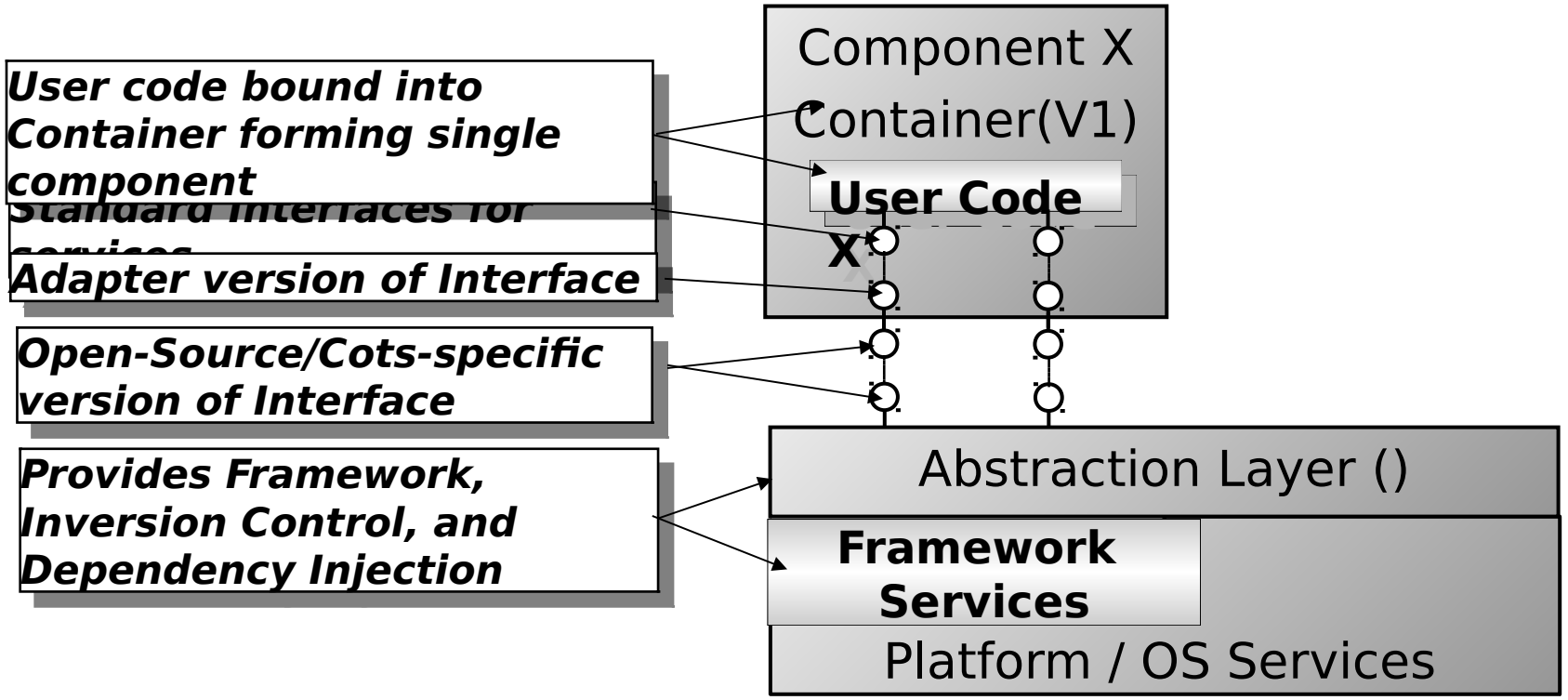
EXPERIENCE. RESULTS.

Example: One way of accessing a WSDL description invoking a SOAP-based Dialog



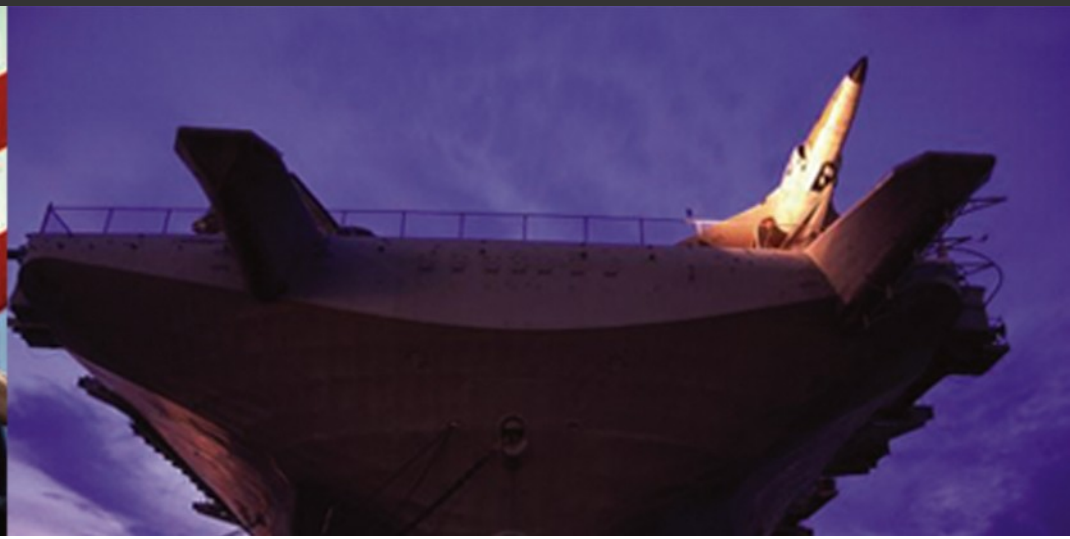
EXPERIENCE. RESULTS.

Example: Using Container Abstraction to implement loose coupling and cohesion



How do SOA's Work: SOA & Security

What should exist in a working SOA



EXPERIENCE. RESULTS.

CSC
EXPERIENCE. RESULTS.

Classic Elements plus new SOA Issues

Classic Security Elements

- Authentication
- Authorization
- Non-Repudiation
- Confidentiality
- Identity Management
- Subjects- Separation of Duties

New Issues

- Policy-Based
- XML-SOAP Messages
- Cross Platform Security Consistency
- Transaction Management
- New Threats
- New Standards
- Complexity
- Recovery and Resiliency to Attack
- End to End Security Management
- Technologies: Tokens and Labeling, Deep Packet Processing, Packet Processing, etc.

CSC
EXPERIENCE. RESULTS.

SOAP Issues

- Is the incoming SOAP message for a live service (e.g., is the requested service valid?)
- Is the SOAP message and its service request valid (e.g., detect a SOAP request for a valid but not-advertised service)
- Does the SOAP message contain Valid Data (e.g., the data packet is invalid for the requested service)

CSC
EXPERIENCE. RESULTS.

PKI Issues

- Establishment of a trusted Certificate Authority that can validate the consumer (software-based consumer) of a certificate
- Making the generation of certificates on clients easy (challenging because clients are pieces of software)
- Being able to negotiate and exchange security keys between the software client and the registered service
- What is/should-be the relationship to the Federal PKI Effort and support for Federal Bridge Certification Authority (FBCA)
- FBCA raises the question of federated identity and attribute management.

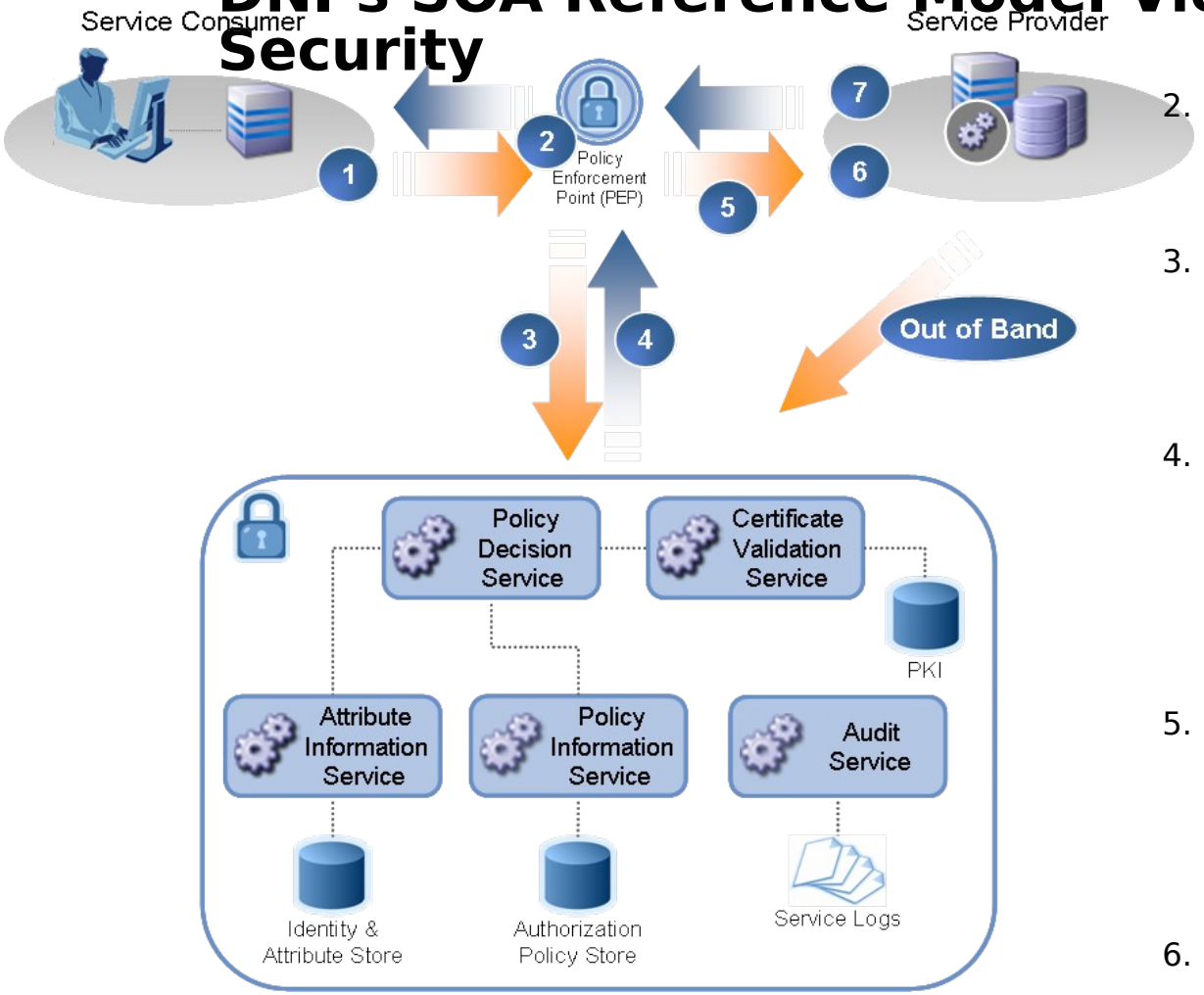
CSC
EXPERIENCE. RESULTS.

Too Many SOA-friendly Security Specifications

- *XML Signature*: A standard specification for a digital signature developed jointly by the W3C and IETF (Internet Engineering Task Force)
- *XML Encryption*: A standard specification developed by the W3C proposes to encrypt portions of XML documents.
- *XML Key Management Specification (XKMS)*: Developed by the W3C to allow clients to obtain cryptographic key information (such as keys and certificates).
- *Security Assertions Markup Language (SAML)*: Defined by OASIS, it outlines a framework for exchanging authentication and authorization information.
- *XML Access Control Markup Language (XACML)*: The primary goal of this specification is to standardize access control language in XML syntax.
- *WS-Security*: The OASIS Web Services Security specification defines a SOAP extension that provides quality of protection through message integrity,

EXPERIENCE. RESULTS.

DNI's SOA Reference Model View of Security



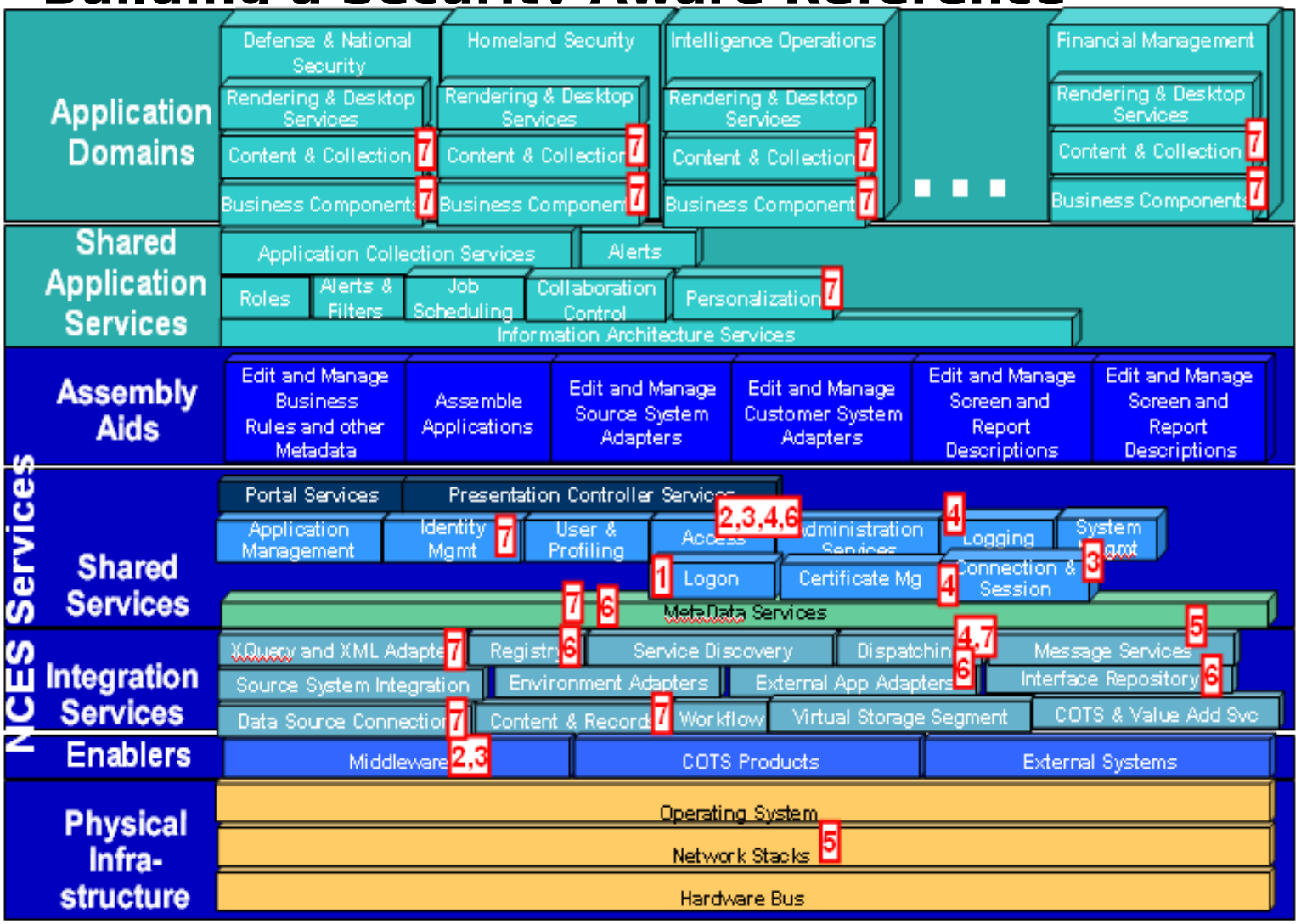
Unclassified//FOUO

Source: DNI's Intelligence Community Service-Oriented Architecture Reference Model , V 2006v01-cd

1. The service consumer agent generates a request message to be sent to the service provider agent.
2. The Security Policy Enforcement Point (PEP) intercepts the message and processes security-related information (e.g. timestamp freshness).
3. The PEP generates a request message to the Policy Decision Service (PDS) to authorize access to the service provider agent for the service consumer.
4. The PDS evaluates, and responds with, a decision that is based upon retrieving attribute information about the service consumer and service provider agent, as well as the authorization policy that exists for the service provider agent.
5. The PEP enforces the decision returned by the Policy Decision Service. If the decision enforced by the PEP is PERMIT, the PEP passes the original request message to the service provider agent.
6. The service provider agent receives the request message and processes it per its business rules.
7. The service provider agent generates an appropriate response message and sends it to the service consumer.

EXPERIENCE. RESULTS.

Building a Security Aware Reference



1,2, …, 7 = Security Service Protection points for a SOA

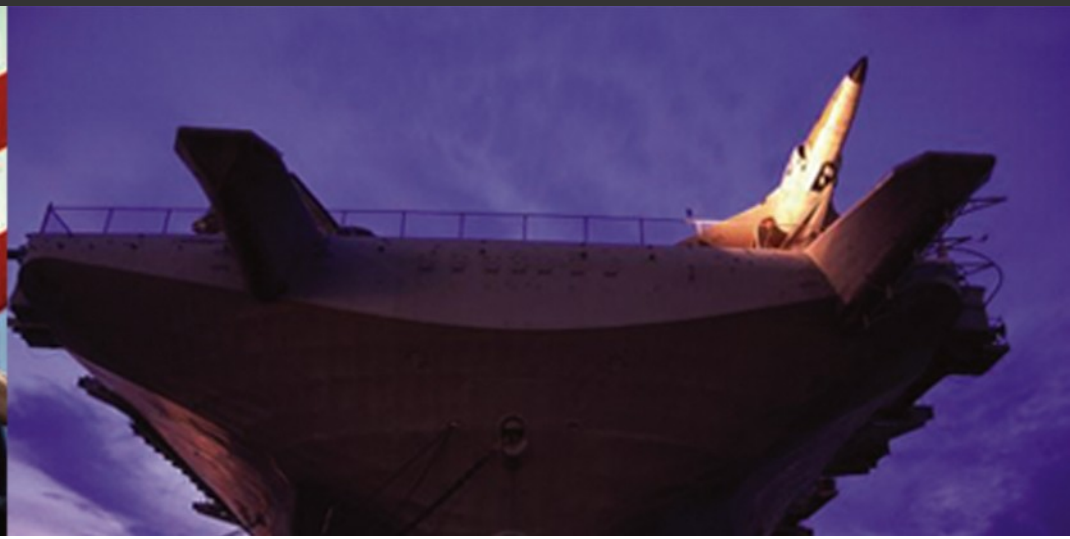
CSC
EXPERIENCE. RESULTS.

Building in Protection Points into the Reference Architecture

PP	Threats
1	User impersonation User exceeding assigned authorization
2	Undesired use of an object implementation Request/response repudiation Disclosure of “eyes-only” data
3	Unprotected security-unaware applications Unwanted revelation of Client machine existence
4	Object masquerade Client masquerade Object misuse of user authorizations Inter-operable Object Reference (IOR) tempering Disclosure of request contents Modification/destruction of request contents
5	Network eavesdropping Message tampering Inability to cross network boundaries (e.g., firewalls)
6	Unprotected security-unaware applications Too many object interfaces and implementation to manage individually
7	Unauthorized disclosure of specific information to client Request/response repudiation Protection of “eyes-only” data

How SOA relates to DoD Architectures and Initiatives

The Distinction between Views, SOA Concepts, and Architecture Products



CSC

EXPERIENCE. RESULTS.

EXPERIENCE. RESULTS.

Relating Initiatives and Architectures: Views, Concepts and Architecture Products

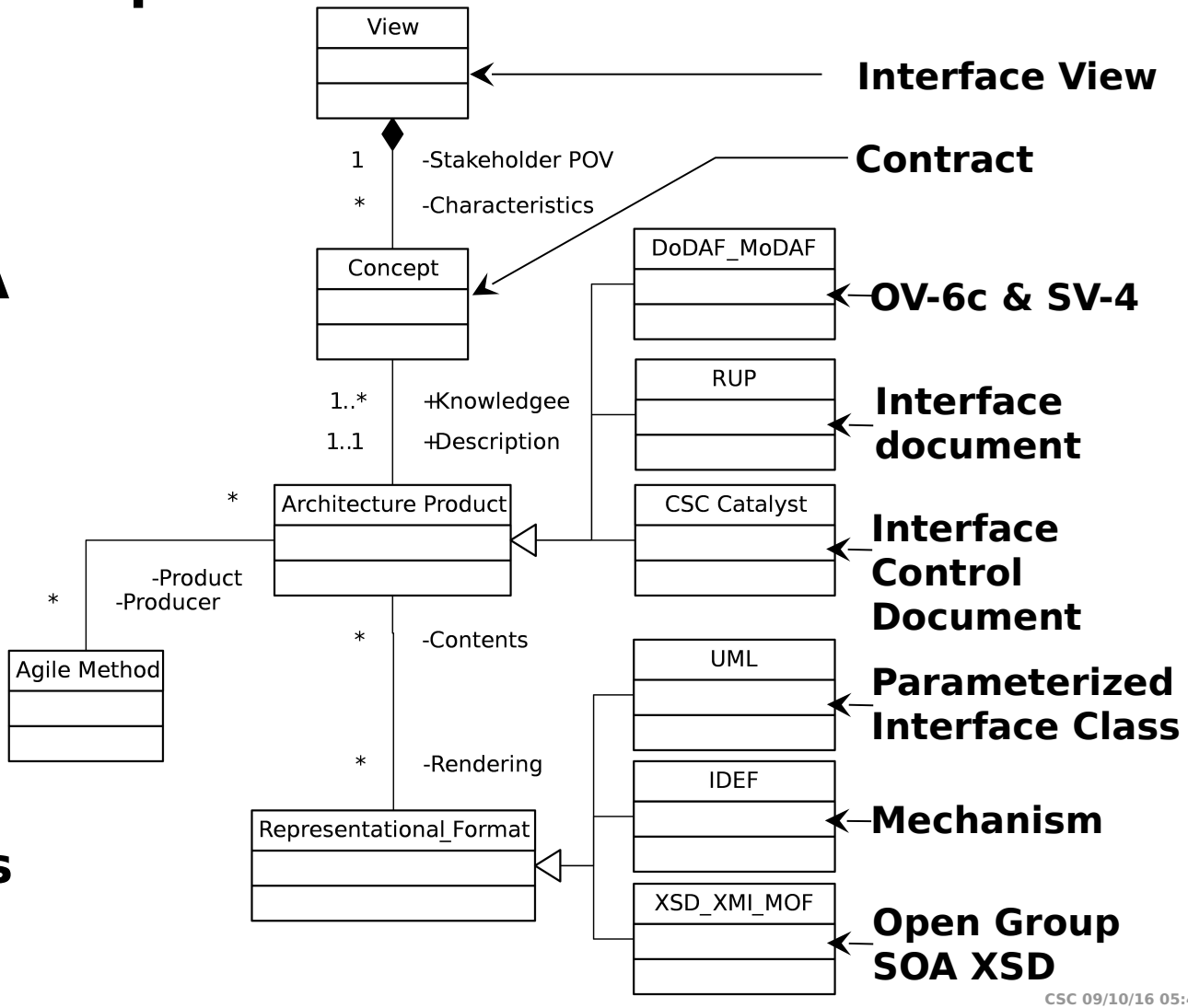
OMG FEA and FTF UML Profiles

OMG/OASIS SOA UML Profile + Ontology

OMG DoDAF/ MoDAF UML Profile

SCRUM, FDD, Army SOA Method

UML 2.1 Profiles



EXPERIENCE. RESULTS.

DoDAF has to be stretched to include everything necessary for SOA definition

Normal DODAF

Service Extensions

CONOP

OV-1
High-Level
Operational Concept

OV-5
Activity Model

OV-2
Operational Node
Description

What to do

SV-4
Systems
Functionality
Description

SV-10c
System Event/Trace
Description

Attributes & Performance Expectations

Industry Best Practice

Attributes and Principles

Service-Based Patterns

Service Repository

How formed and
How to Behave

Service Specification

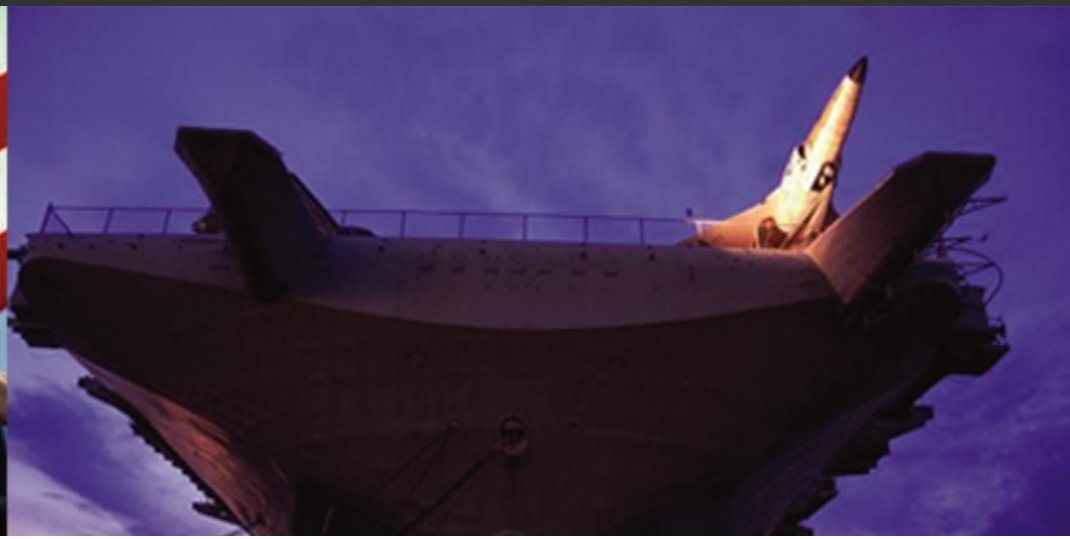
EXPERIENCE. RESULTS.

Some interesting Initiatives

- One classified program is for an IC Agency is completely replacing existing systems
- DISA's Net-Centric Enterprise Service
  - The concept of managed services using a common and uniform set of foundational services
- Navy SPAWAR
  - Establishing a 'community of practice' approach first

Lessons Learned

Some from the School of Hard Knocks



CSC

EXPERIENCE. RESULTS.

EXPERIENCE. RESULTS.

SOA Systems Engineering: Not the same as Traditional

SE	Traditional SE	SOA SE
Delivery	System = Program = Silo	System = Capability = reused set of services
Delivery Organizations	Traditional Program Team of Developers	• Component Developers • Business Model Assemblers • Application Assemblers • Librarians
Methodologies	Structured Analysis & Design OOAD can be used if desired	Lightweight Methodologies and processes required Crystal, FDD) OOAD an assumed foundation
SDLC	Waterfall, Incremental, or Prototype based	Iterative (not Incremental), continuous, evolutionary, SCRUM or similar based
Architecture Engineering	Structured Decomposition DFD based	Interacting View-Based (e.g., Model Driven Architecture, View-based, High degree of semantic control)
Integration	Point-to-Point	Dynamic and Static Interfaces Discovery Mechanisms
Verification	Issues with major conceptual changes between phases	Iterative addition of Detail

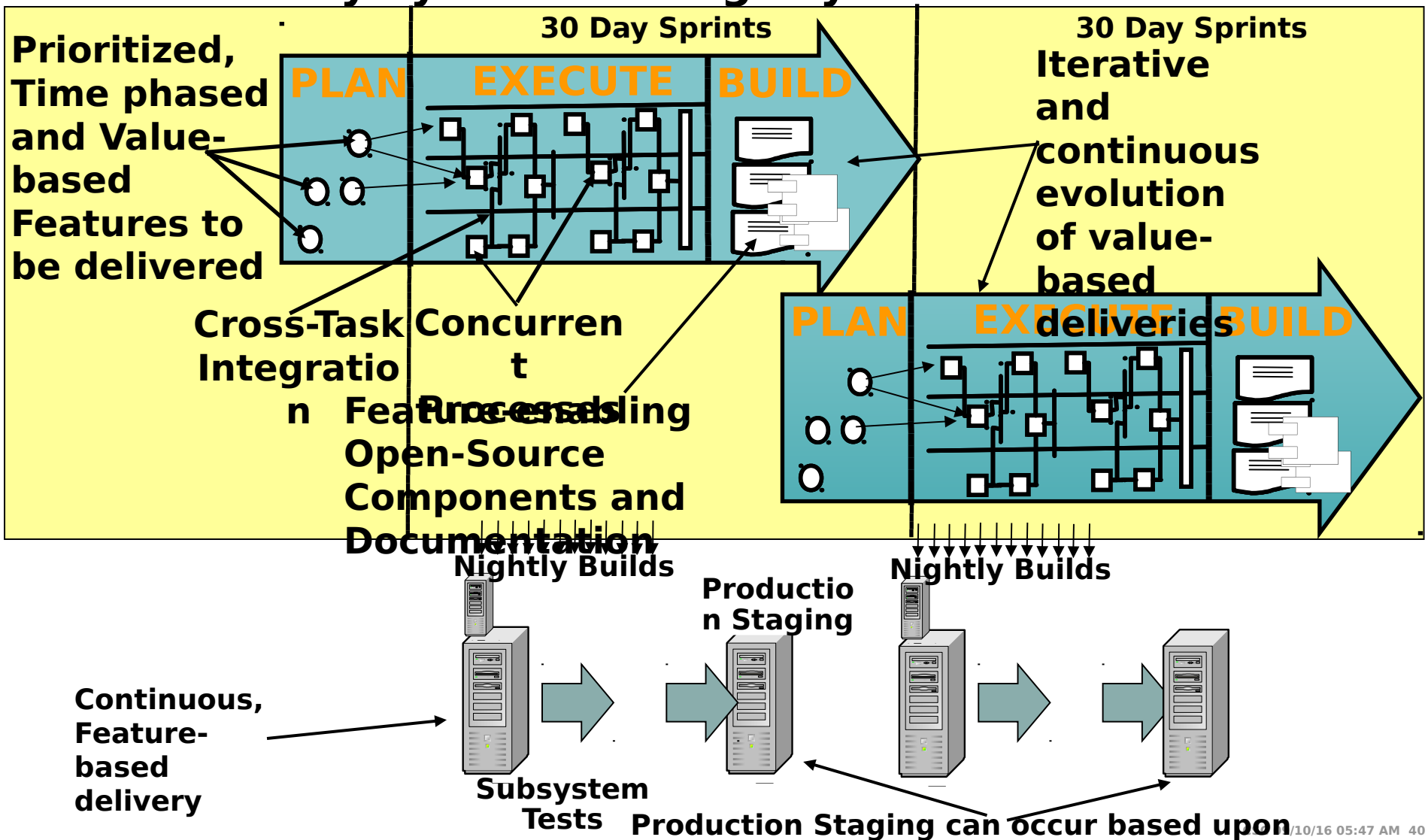
EXPERIENCE. RESULTS.

SOA Systems Engineering: Agility means delivering features more quickly than the requirements change cycle

- Requires Software Systems Engineering methods and processes
- Plan and deliver by feature, not program or program increment
- Use lightweight methodologies (e.g., FDD)
- Use continuous delivery processes (e.g., SCRUM or Crystal)

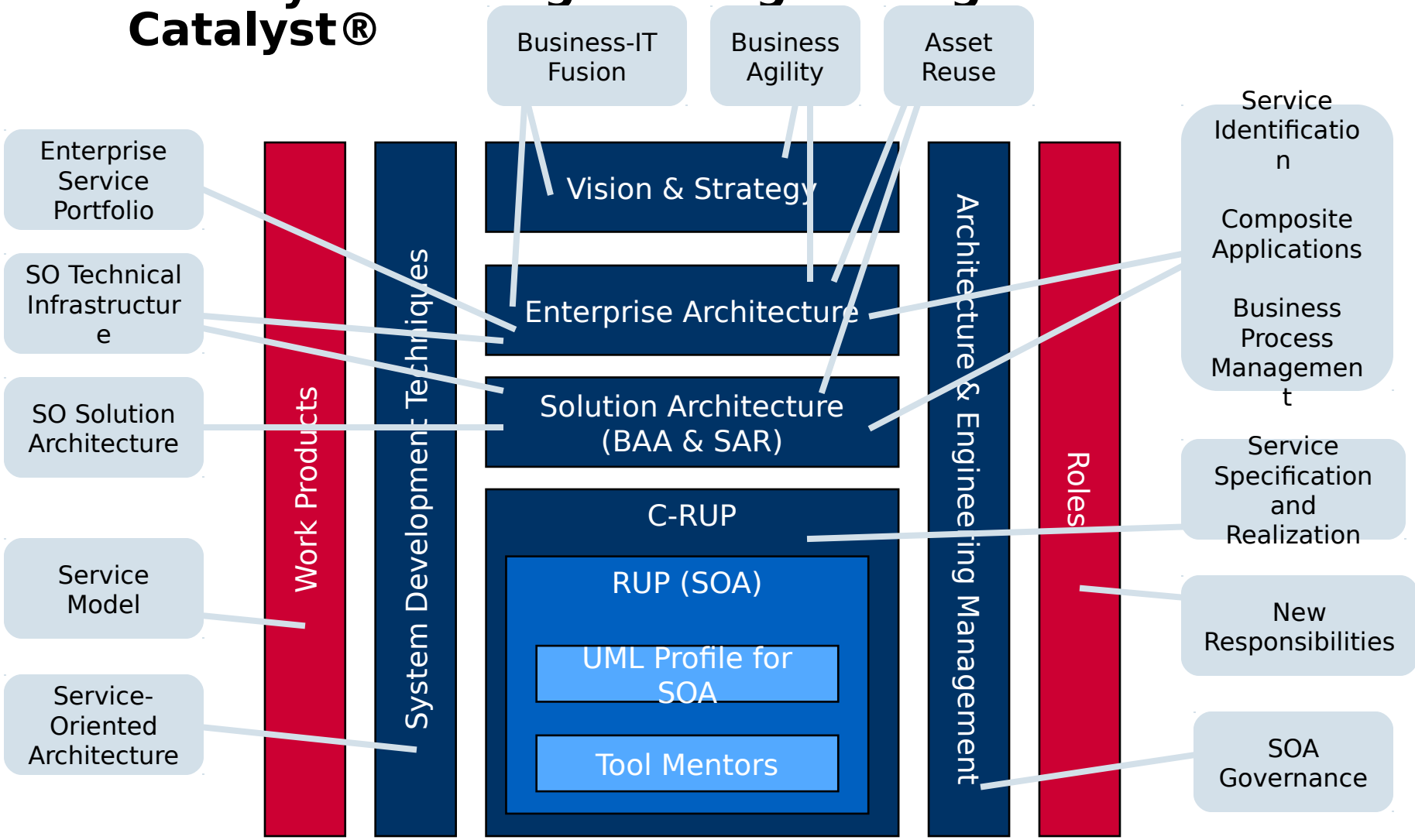
CSC
EXPERIENCE. RESULTS.

SOA Systems Engineering: Example--30 day delivery cycles with nightly builds and



EXPERIENCE. RESULTS.

SOA Systems Engineering: Changes to CSC's Catalyst®



EXPERIENCE. RESULTS.

Interoperability & NetCentricity – Plan for multiple levels

- Interoperable information sharing is a set of enterprise characteristics
- These characteristics are additive: Level 5 interoperability contains Level 4 and Level 4 contains Level 3 …
- Interoperable Information Sharing starts with achieving level 1 and continues to level 5.
- Level 5 can access level 4,3,2, or 1 but lower levels can only access higher levels using that level's characteristics

Level 5
Dynamic and Discoverable
- Context independence
- Constraints on how Components are built
- Common metadata registry federated across domains

Level 4
Interoperable Data Exchange
- Identity Mgmt; Data Exchange; Method Exchange
- Interface Registration Constraints
- Bus-Based Implementation
- Interface Payload Control
- Containers, Connectors

Level 3
Interconnected (All types)
- Local Service Registry becomes Service Proxy
- Hub-based implementation (Runtime Interface Registry; Dispatcher)
- Full SOA Design and Runtime Documentation Required

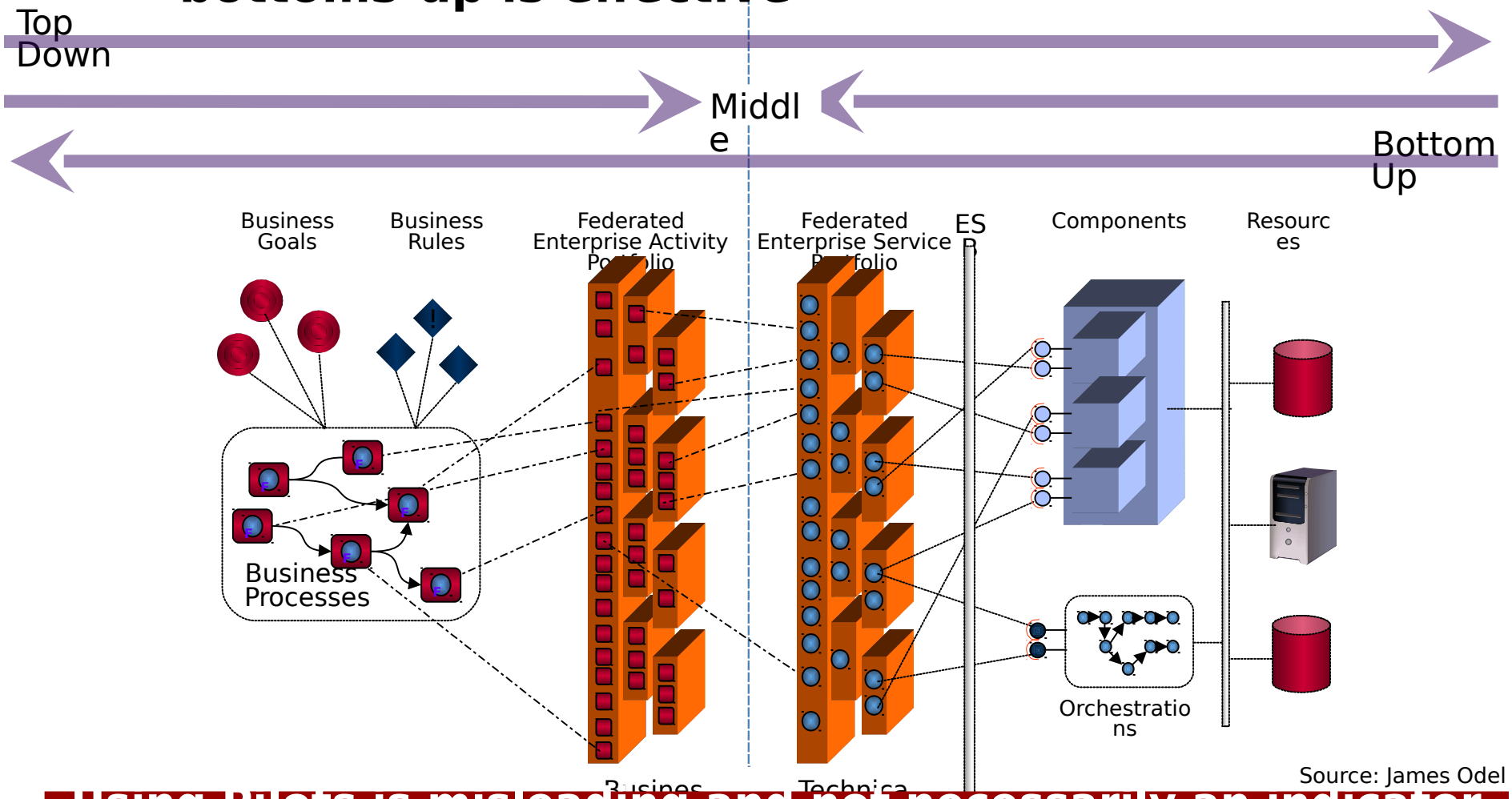
Level 2
Controlled
- Still point-to-point
- No hard-coded interface connections
- Common Local Service Registry as a config. file
- Common Interface Design time documentation
- Basic Design time information

Level 1
Specified
- Common Specification Format
- Control over the Service Components Built
- Basic Design-time Documentation
- No control over runtime implementation
- Point-to-point, program-to-program
- Data usage context specified in metadata

Level 0
Little or no integrated information

EXPERIENCE. RESULTS.

SOA Adoption: Top-down is efficient / bottoms-up is effective

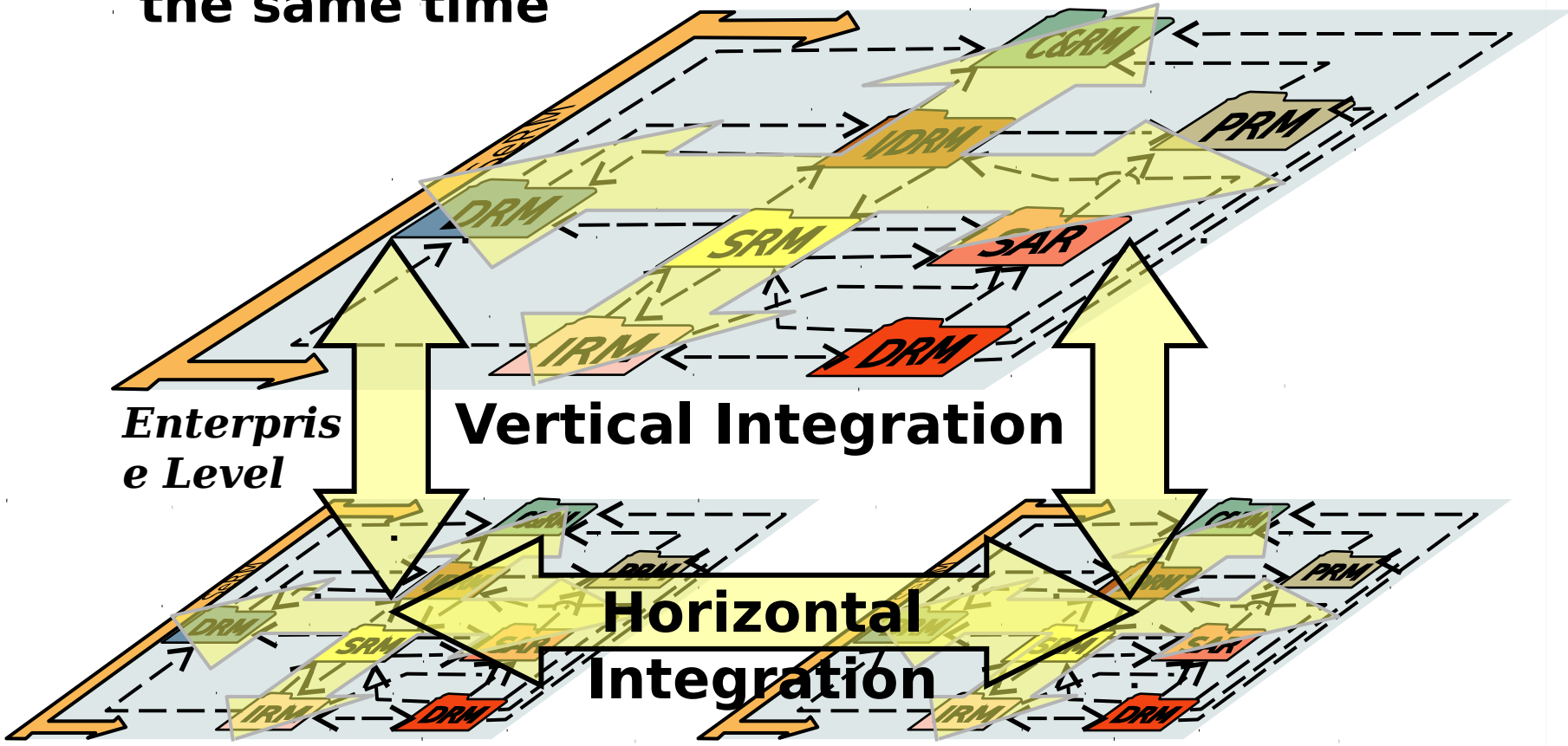


Source: James Odel

Using Pilots is misleading and not necessarily an indicator of success

CSC
EXPERIENCE. RESULTS.

SOA Adoption: Reality – you have to do both at the same time



You have to have a master blueprint of what is going to be delivered and how they fit together.

EXPERIENCE. RESULTS.

CSC's STORM (Solution Template and Open-Sourced Resource Model) Framework

Challenges

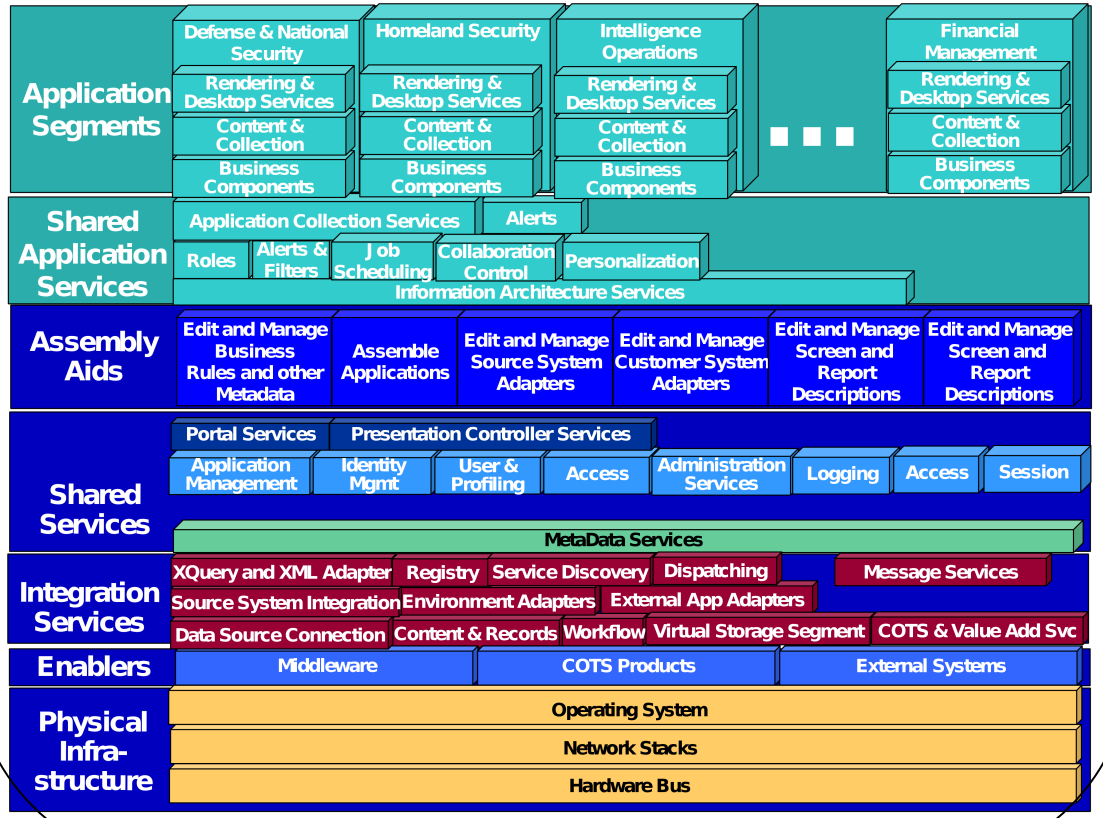
- **Continuously deliver features with agility**
- **Time to Market critical**
- **Concurrent delivery of autonomous services**
- **Reuse Common Assets Enterprise Services**

Lessons Learned:

- Used to initialize design/delivery for systems (e.g., Air Force C4 systems).
- Open-Specification with reference code
- Showed that a framework can be used as the basis for extension and specialization
- Metadata driven approaches make application context binding simple
- **NOTE:** 88% of construction involved architecting, building and testing the IF services

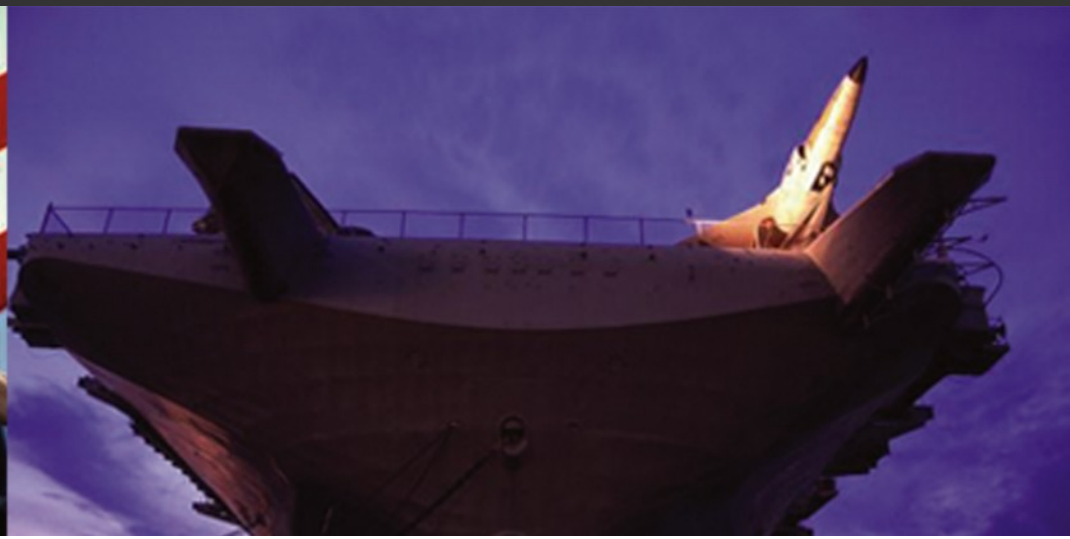
CSC Solution

CSC's STORM Framework Used to initialize SOA Design and Delivery for a Number of



Skills and Roles

Delivery of SOA depends upon matching the skillset to the roles required



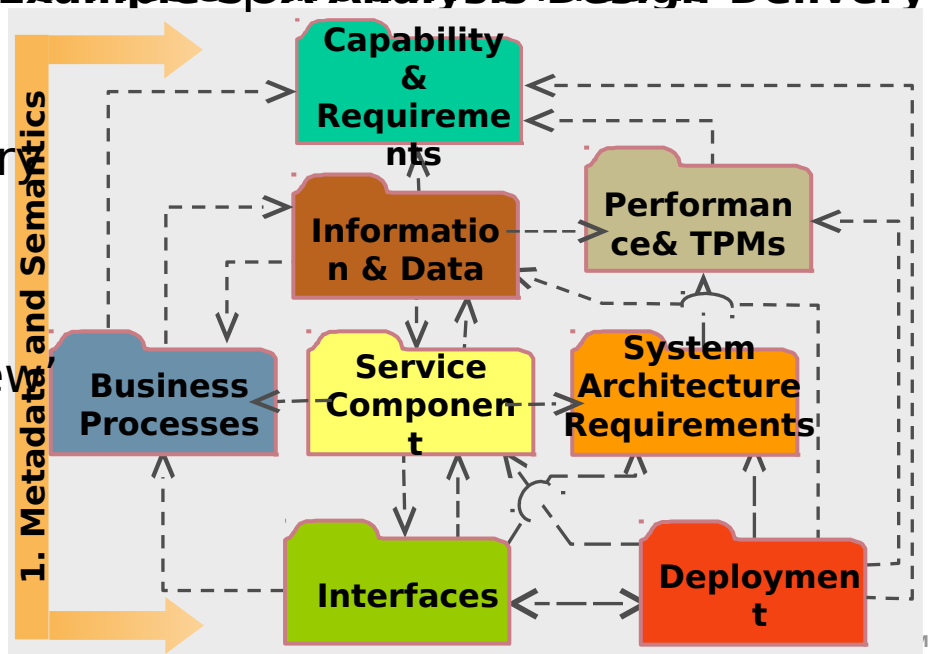
EXPERIENCE. RESULTS.

EXPERIENCE. RESULTS.

Skills are dependent upon what Needs to be produced

- Paradox 1: as agility of the deliverable increases so does the formality and rigor of the software systems engineering
- Paradox 2: as software systems engineering formality increases so does the need to for flat delivery organizations divided into small teams that execute semi-autonomously
- To staff a SOA and resolve these paradoxes means —
  - Utilizing a more formal Analysis-Design-Delivery Model
  - Skills are selected and organized around the needs of each SOA 'View'

Example SOA Analysis-Design-Delivery



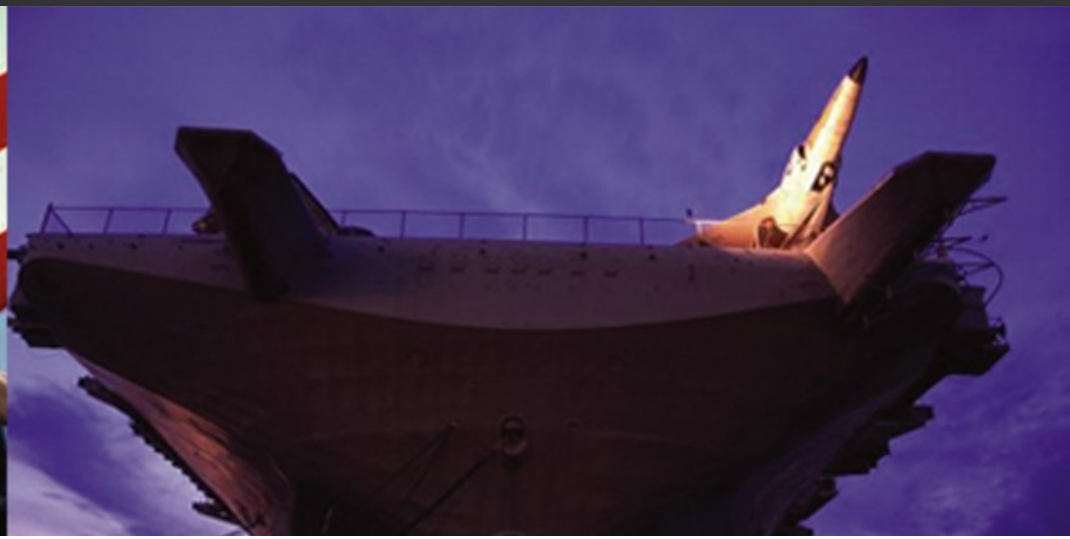
EXPERIENCE. RESULTS.

Matching the Deliverable to the Role and Skills

Deliverable	Role	Skill Notes
1.PMP	Program Manager	Must understand continuous & Concurrent Delivery
2. Strategy	Chief Architect	Trained in application of SOA principles to problem
3. Processes & Systems Engineering	Chief Engineer	Continuous and concurrent engineering processes
4.Acceptance	Validation Engineer	ITIL trained and capable
5. Quality	QA-Business QA-Technical PRM Engineer	Focus on orchestration and traceabilty Focus on pattern selection and principle usage Focus on SLA to metrics to pattern transformation
5. Business/ Mission Processes	Domain Expert Requirements & Measurements	Create measurable Capability Model SLAs; Allocate Requirements to capabilities
	Sr. Analysts	Business Process modelers
6. Architecture:: Services	Service Component Engineers	Design the class and collaboration models
	Service Interface Engineers	Designs the interface model
7. Architecture:: Static Security	Sr. Security Architect	Protection points, security containers
8. Architecture: Dynamic Security	Sr. Security Architect	Message security, COI, Context
8. Architecture:: Data	Information & Data	Service-based data partition design

The Future

Convergence around an integrated model

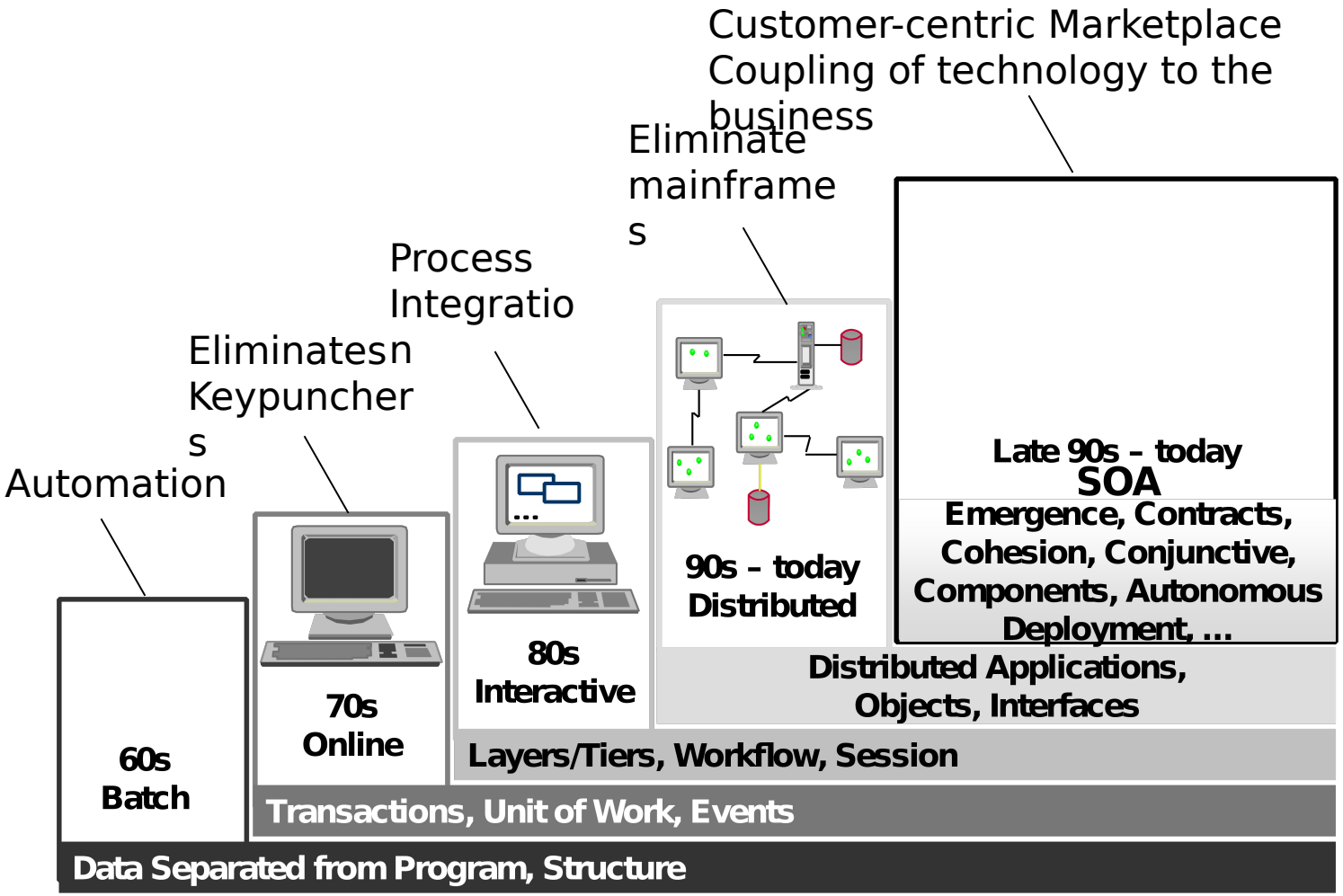


EXPERIENCE. RESULTS.

CSC
EXPERIENCE. RESULTS.

Will SOA be around in ten years?

- Yes, because it is the first technical advance that couples business/mission goals/objectives with technical capability



EXPERIENCE. RESULTS.

Some Predictions

From the Professional Prognosticators

- $3 Billion spent on failed SOAs — Gartner
- 60% of Enterprises will use SOA by 2008 — Gartner
- SOA Governance a Key and starts to emerge as a first-order problem — Gartner, Forester, …

Convergence on what a SOA should contain

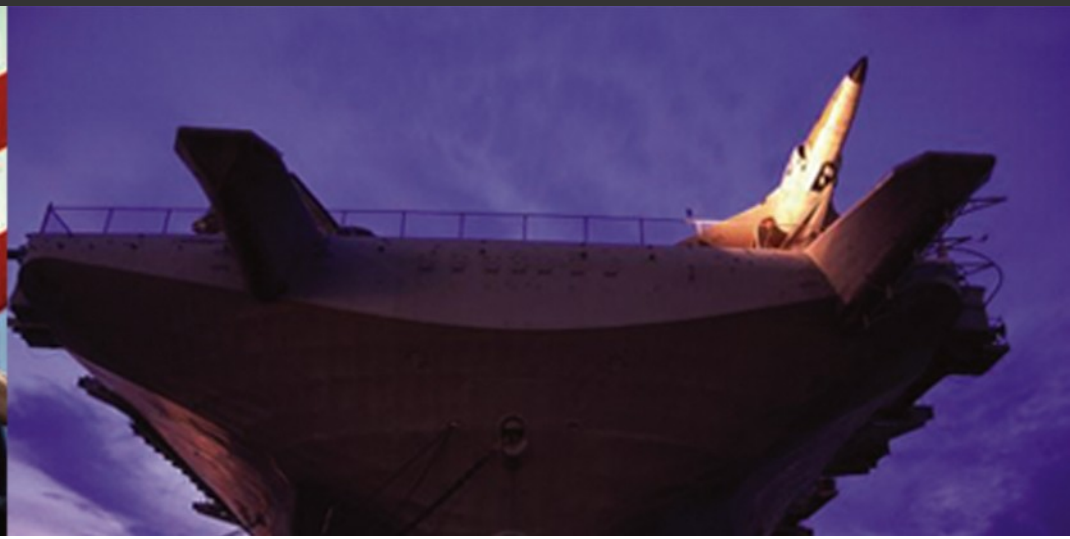
- OMG: specifying a UML Profile for SOA
- OASIS/OMG Ontology convergence

Improved Tool Support

- SPARX EA, IBM-Rational RSA, and Telelogic Rhapsody will adopt to SOA UML Profile
- Model-driven approaches for generating into specific busses (e.g., Weblogic, Websphere, Tibco) mature
- The Open-Source SOA market will either follow SUN this year, or fold

The End

SERVICE-ORIENTED ARCHITECTURES
An Overview



EXPERIENCE. RESULTS.